Università degli Studi di Milano

Ph.D. Program in Computer Science

(XXXII Cycle)

Department of Computer Science

A thesis submitted for the degree of

Doctor of Philosophy

Algorithms, Learning, and Optimization

Author
Tommaso R. Cesari

Supervisor
Nicolò Cesa-Bianchi

PhD Coordinator
Paolo Boldi

Academic Year 2018–2019

Contents

Introduction **5**

1 Dynamic Pricing **11**
1.1 Introduction . . . 11
1.2 Further related works . . . 13
1.3 Preliminaries and definitions . . . 13
1.4 Lower bounds . . . 14
1.5 Distribution-free bounds . . . 15
1.6 Distribution-dependent bounds . . . 18
1.7 Conclusions . . . 23
1.8 Deferred proofs and additional results . . . 23
1.8.1 Lower Bound . . . 23
1.8.2 Cautious search . . . 27
1.8.3 UCB with inflated confidence bounds . . . 28
1.8.4 Two valuations . . . 30
1.8.5 Nonstochastic dynamic pricing: some initial results . . . 35

2 Cooperative Online Learning **39**
2.1 Introduction . . . 39
2.2 Related works . . . 40
2.3 Preliminaries and definitions . . . 41
2.4 Online Mirror Descent . . . 42
2.5 Stochastic activations . . . 43
2.6 Lower bound for stochastic activations . . . 45
2.7 Nonstochastic activations . . . 46
2.8 Conclusions . . . 48
2.9 Deferred proofs on stochastic activations: multiple agents . . . 48

3 Online non-Concave Maximization **51**
3.1 Introduction . . . 51
3.1.1 Main contributions . . . 52
3.1.2 Earlier works: some further comments . . . 53
3.1.3 Outline of the chapter . . . 54
3.2 Assumption, definitions, and notation . . . 54
3.2.1 Assumption on f . . . 54
3.2.2 Useful notation and definitions . . . 55
3.2.3 Sets of near-optimal points, with examples . . . 56
3.3 Deterministic perturbations . . . 58
3.3.1 Budget on queries . . . 58
3.3.2 Automatic stopping . . . 62

3.4 Stochastic perturbations 65
3.5 Deferred proofs and additional results 67
3.5.1 Useful inequalities about packing and covering numbers 67
3.5.2 Simple bound on the near-optimality dimension 68
3.5.3 Proofs of Corollaries 3.1 and 3.3 69
3.5.4 Connections with a known regret bound in dimension one 70

4 Repeated A/B testing **73**
4.1 Introduction and related work 73
4.2 Problem definition and notation 75
4.3 Choice of performance measure 76
4.4 Explore then exploit 78
4.5 Conclusions and future work 81

Abstract

This thesis covers some algorithmic aspects of online machine learning and optimization. In Chapter 1 we design algorithms with state-of-the-art regret guarantees for the problem dynamic pricing. In Chapter 2 we move on to an asynchronous online learning setting in which only some of the agents in the network are active at each time step. We show that when information is shared among neighbors, knowledge about the graph structure might have a significantly different impact on learning rates depending on how agents are activated. In Chapter 3 we investigate the online problem of multivariate non-concave maximization under weak assumptions on the regularity of the objective function. In Chapter 4 we introduce a new performance measure and design an efficient algorithm to learn optimal policies in repeated A/B testing.

Acknowledgments

The author gratefully acknowledge the support by Criteo AI Lab, COST Action CA16228 “European Network for Game Theory” (GAMENET), and the Google Focused Award “Algorithms and Learning for AI” (ALL4AI).

Moreover, the author sincerely thanks his supervisor Nicolò for his continuous support, invaluable mentorship and overall awesomeness!

Introduction

Online learning is a sub-field of machine learning in which information becomes available sequentially [82]. After every new piece of data has been acquired, the learner is asked to make a prediction based on the history observed so far. This is in contrast to batch learning techniques in which the entire training data set can be accessed all at once [84].

The usefulness of online learning algorithms has been long established. Even with today's technology, the growing popularity of "big data" makes batch learning often unfeasible. Moreover, an online approach is necessary to adapt to evolving patterns in the data. This happens whenever new data is generated over time. For example: sensor data, financial data, user interaction data, and so on. In these cases, traditional learning protocols where predictors are generated by feeding a fixed-size training set to a learning algorithm, become inefficient. This happens because every time new training data are available one would have to run the algorithm again from scratch.

The field of *online convex optimization* [42] is a prominent example of the online learning paradigm. Its protocol proceeds in time steps. At each time step, the learner picks a *decision*, that is modelled by an element of a convex subset of $\mathbb{R}^d$, then suffers a loss determined by the decision. Losses are real functions defined on the decision set that can vary over time steps, and are generally assumed to be bounded. After the player suffers a loss, the loss function is revealed. An algorithm for online convex optimization that runs for T time steps outputs a sequence of T decisions, with the goal of minimizing the *regret*. The regret is defined as the difference between the cumulative loss of the decisions picked by the algorithm and the cumulative loss of the best decision in hindsight. One of the main reasons why online convex optimization has become one of the most important online learning framework is because of its broad set of applications, such as online routing, ad selection from search engines, spam filtering, etc [42].

An important special case of online convex optimization is the problem of learning with expert advice. In this setting, the decision set is the probability simplex over a finite set of elements typically referred to as *experts*. By defining a loss for each expert, we can define the loss of all these distributions x as the expectation of the loss of a random expert (drawn according to x). Losses of experts are usually assumed to be bounded in $[0, 1]$. The reason why this setting is particularly important is that in many real-life applications, ranging from weather forecast to stock-price prediction, the choice is indeed limited to a finite number of options, and after following the advice of an expert it is possible to measure how good the advice of the other experts really was (as in the examples mentioned above).

There are scenarios, however, where this type of feedback might not be available. Consider for example the problem of placing an ad on a web page picked out a finite set of ads. After choosing one, it is possible to determine how good the choice turned out to be but it is not possible to know how good a different ad would have been, had it been chosen instead. This type of setting in which only the loss of the decisions that are picked are revealed to the learner is called online learning with *bandit* feedback, and it covers an array of settings that prediction with expert advice does not. Besides ad placement, a concrete example of this scenario, and one of the main topics of this dissertation (Chapter 1, a joint work with Nicolò Cesa-Bianchi and Vianney Perchet [25]), is the so-called *dynamic pricing* problem. In dynamic pricing (also called posted price auctions) a buyer with an unlimited amount of identical goods interacts sequentially with a series of buyers. During each interaction the seller offers to sell the good at a given price, that can (and should!) be adapted dynamically over time. Each buyer has a privately held valuation of the item on sale, and buys it if and only if the proposed price is lower than this valuation. The goal of the seller is to minimize the *regret*, defined as the difference between the cumulative reward of the best fixed price in hindsight and the total reward accrued by proposing a sequence of prices. We study the case of a segmented market, i.e., when the population is partitioned into an unknown number of latent types characterized by their own private values, and buyers arrive according to i.i.d. draws from an unknown fixed distribution over these types. Incidentally, this setting also applies to repeated second-price auctions where there is only one relevant buyer. (Auctions in general —and second-price auctions in particular— are receiving more and more attention due to their application to *real-time bidding*, a process that regulates the vast majority of the online advertisement business.)

A key quantity for dynamic pricing is the so called *demand curve*, that is the function mapping each price to the probability of selling the good at that price. Typically, the demand is assumed to be at least continuous [52]. In a segmented market, however, the function presents an unknown number of discontinuities and sharp drops after each one of them. While these discontinuities prevent us from leveraging usual learning techniques, the simple piecewise constant form of the demand curve could in principle be of help in the analysis of the problem.

Note that on one hand the seller has to figure out the best price among a continuum of options, but on the other hand it would be desirable to have bounds that scale with the unknown number of different private values, which are the only potentially optimal prices. Notably, if those values were known in advance, the problem would reduce to a standard multiarmed bandit problem. Not being so, these potentially optimal arms have to be located *during* the learning process. This extra layer of uncertainty is one of the features that distinguishes this problem from standard bandit settings.

An accurate implementation of a search procedure has to be made to address the issue. The most natural type of search, a binary search, turns out to be suboptimal. To see why, assume for now that the seller only has to locate a certain private value w having access to an oracle that returns the expected payoff of each price. Naively, one might want to minimize the number of evaluations of the payoff function (or equivalently, of the demand curve) to get ε-close to a maximizer of the function. This would be good enough to ensure an ε-optimal price because the payoff function is 1-Lipschitz in the worst case. The issue with using a binary search approach for this task is that it does not take into account the specific structure of the problem. Specifically, offering a price that is slightly higher than w is significantly worse than offering a price that is slightly lower. Keeping this in mind, we adapt a technique first appeared in [52] for the special case of one single hidden type that we call *cautious search*. The intuition behind cautious searches is that for regret minimization one should not simply focus on minimizing the number of rounds needed before approaching w. It is more costly to take less time and sell less goods than to take longer while making numerous sales along the way. We present an algorithm that combines this cautious search idea over specific subintervals of price values with a UCB-like procedure that steers the exploration towards the most promising areas.

Another technique that we need is a way to discover all hidden types without knowing their number beforehand. The idea is the following. At the beginning, the algorithm tries to locate the smallest private value by determining if there is a significant difference in demands of successive posted prices. As soon as one is spotted, the initial interval is split into two subintervals, trimming down suboptimal areas. After that, the UCB-like algorithm overseeing the process determines the current most promising interval, ad the process moves forward similarly. This way the algorithm finds all the (promising) valuations adaptively while current best intervals are explored and refined.

The literature on discontinuous demand curves is significantly more limited than that of continuous ones. There is however a recent result that could be applied to our setting and we can use as a benchmark [30]. It requires the knowledge of the smallest drop γ in the demand (i.e., the smallest probability of a type of buyers) as well as the number of types K. It gives a regret bound of order at best $(K^{12}/\gamma^8)\sqrt{T}$, where T is the time horizon. In contrast, our algorithm do not require the knowledge of K and enjoys a regret of order at worst $\big(1/\Delta + (\log\log T)/\gamma^2\big)(K \log T)$, where Δ is the gap between the revenue of the optimal type and that of the second-best. Notably, this can be reduced to $(\log T)(\log\log T)/\Delta$ with no prior information on γ in the special case of two hidden types. We also present an algorithm attaining a regret bound of order $\sqrt{KT}$ independently of the distribution over types and we show that for distribution-free bounds, this rate is unimprovable. We conclude the chapter by giving some preliminary results and ideas on the non-stochastic case.

Some of the technical issues that arose from the study of this dynamic pricing setting are due to the problem being inherently non-convex. This prevents the use of usual search procedures to locate the optimal prices. It is worth noting that a recent work studied the connections between some dynamic pricing settings and online convex optimization [65]. Designing such reductions allows to focus on the study of specific aspects of the already existing online convex optimization arsenal that might be relevant to other problems. For example: in a world where distributed systems are ubiquitous, is worth investigating how these online convex optimization techniques behave in a cooperative framework. For these reasons, in Chapter 2 we consider the problem of cooperative online convex optimization, also known as cooperative online learning with full information feedback. (This is a joint work with Nicolò Cesa-Bianchi and Claire Monteleoni [27].) In this setting, a network of agents is trying to minimize a common long-term objective. At each time step, a subset of agents is activated and asked to make predictions. Then, the system is charged with the average loss of the predictions of the active agents and the loss function is revealed to them and their neighbors. The goal is to minimize the network regret, defined as the difference between the cumulative regret accrued by the system and the total loss of the best fixed action in hindsight.

This setting is motivated by mobile systems cooperating towards a common goal, in which battery life or short-range communication constrain the free flow of information. Concretely, this kind of techniques have been applied to distributed environmental monitoring, where they empirically showed improved performances compared to non-cooperation versions [62, 63]. However, a theoretical analysis of the phenomenon was missing.

We investigate if and to what extent knowledge about the network topology helps in speeding up learning when information is shared. If agents activations are stochastic, we show that communication helps greatly and no information on the graph structure is needed to perform optimally. In fact, if all agents run the same vanilla version of Online Mirror Descent, with the same initialization, and updating their local model whenever they get a chance to do so, the system regret is of order $\sqrt{\alpha T}$, where α is the independence number of the communication network. We then prove a lower bound showing that this rate cannot be improved upon, even if agents have complete information about the graph. This is in general significantly better than $\sqrt{NT}$, where N is the number of agents, which is the regret rate that the system would achieve if agents behaved independently, only making updates when they were activated. If activations are chosen adversarially, the situation changes completely. We show that for some graphs there is an oblivious choice of losses and activations for which *any* algorithm that ignores the graph structure incurs a *linear* regret, making learning impossible. This striking difference depends on the fact that when activations are non-stochastic the adversary has the power to amplify the spread of *bad* information and reduce that of good feedback. In this case agents are better off learning by themselves. However, we show that this lower bound can be broken if agents do have information about the graph structure at least at their initialization. If the network is partitioned into cliques and agents ignore all feedback coming from outside of their clique, the regrets scales with $\sqrt{\overline{\chi} T}$, where $\overline{\chi}$ is the clique-covering number of the communication network.

These results focus on the communication part of learning. At their core rest single-agent online convex optimization algorithms with losses that vary over time. A special case of such single-agent settings is when losses are *independent* of time. In other words, when there is one single loss function ℓ and the goal is to find an approximate minimizer of ℓ as quickly as possible. The same problem is sometimes (equivalently) formulated for maximization of concave reward functions. As we mentioned above, a vast literature is devoted to these types of problems due to their numerous applications. However, there are many other reasons for investigating its non-covex/concave conterpart. They are not simply important as a theoretical tool but they find a vast number of applications on signal processing, bio-informatics, and machine learning (deep learning in particular), just to name a few [47]. For these reasons, in Chapter 3 we drop all convexity/concavity assumptions and move on to studying how to determine approximate optimizers of multivariate functions. (This is a joint work with Clément Bouttier and Sébastien Gerchinovitz.[1]) We will keep the focus on the online setting, where the values of the objective function can be only be accessed sequentially, one at a time. Several issues appear as an immediate consequence of dropping the concavity assumption:

[1]An earlier version of this work appeared in [13]. The setting has now been extended to functions defined on general bounded d-dimensional domains, and the deterministic analysis now holds for adversarial perturbations. The regret bounds stated in the main theorems are tighter, and proofs have been revisited and significantly simplified.

1. existence of maxima is no longer guaranteed (not even locally);
2. even when a global maximum exists, it is not guaranteed to be unique;
3. even when a unique global maximum exists, algorithms can get stuck on local maxima and never converge to the global optimum;
4. even if they do, the speed of convergence might be significantly slower and harder to quantify;
5. even when they converge quickly to a global optimum, computational costs might severely increase.

All these problems are of great importance both from a mathematical and from a practical point of view. In this dissertation we will mainly focus on items 3 and 4, i.e., on the design and analysis of algorithms that converge quickly to global maximizers.

In order to compensate for the lack of regularity that comes with the loss of concavity, the objective is usually assumed to be globally continuous in some strong sense, typically Lipschitz, and defined on a compact set. We will not make any of these assumptions. We only assume that the domain of the objective is bounded and the function satisfies a weak regularity assumption centered at a maximizer that we call *lipschitzness around a maximizer*. Formally, a function $f\colon D \subset \mathbb{R}^d \to \mathbb{R}$ is L_0-Lipschitz around $x^\star$ with respect to a norm $\|\cdot\|$ if for all $x \in D$, $\big|f(x) - f(x^\star)\big| \le L_0\|x - x^\star\|$. To give some perspective, recall that a Lipshitz function is globally continuous and differentiable almost everywhere on its interior by Rademacher's theorem. In contrast, a function that is Lipschitz around a maximizer can not only be nowhere differentiable on its domain, but even *discontinuous everywhere* but at that maximizer (see Assumption 3.1 and subsequent discussion). To make matters worse, we will assume that the values of the objective cannot be accessed exactly but are subject to perturbations, either deterministic or stochastic.

We extend an old algorithm from the seventies [73, 86], that we call Piyavskii–Shubert algorithm to our non-compact, weakly regular, multidimensional setting with perturbations. We measure the performance of all our algorithms with the simple regret, defined as the difference between the maximum of the objective and the value of the objective at the point returned by the algorithm. When perturbations are deterministic, we first analyze a version of the Piyavskii–Shubert algorithm that has a finite budget of evaluations of the objective function. As soon as this budget is exhausted, it has to halt returning a point. This will serve as a base result to build the rest of the theory. We then move on to algorithms that stop automatically guaranteeing an approximate optimal point after stopping, when perturbations are either deterministic or stochastic. In all cases we bound the number of samples needed to reach any precision ε. These bounds are expressed in terms of the "size" of a decomposition of the domain in sets of suboptimal points. This agrees with the intuition that the bigger the size of these sets (i.e., the more suboptimal points there are), the harder the optimization problem is. As it turns out, the right notion of size is the so-called *packing number* (3.3). To give our results even more concreteness, we then bound these packing numbers in terms of the so called *near-optimality dimension* (3.6), a commonly accepted parameter to measure the hardness of an optimization problem. We prove that when the near-optimality dimension is zero, our algorithms converge to a maximizer exponentially fast in the number of evaluations. Otherwise, they converge as an inverse power of the near-optimality dimension, confirming and quantifying that the smallest the near-optimality dimension, the easiest the optimization problem is (see, e.g., Corollary 3.2).

This non-concave optimization problem can be seen as a bandit setting with a multidimensional continuum of arms whose rewards are given by the objective function. Bandits with infinite arms and variants thereof appear in several fields [22, 89, 90]. As we mentioned above, pricing problems can be seen as instances of such problems. (The the action space is the set of prices and the feedback is given by the earnings of each transaction). An example coming from statistics, and the topic of Chapter 4, is the problem of repeated A/B testing, which in its original formulation dates back to the fifties [91]. (This is a joint work with Nicolò Cesa-Bianchi, Vianney Perchet, and Yishay Mansour [26].) A new wave of interest has gone to this problem in recent years due to the rise of online advertising companies. As a concrete example, consider one of such companies that profits by selling ads online according to some specific technology. Periodically, the research team of the company will come up with new solutions with the goal of increasing the profit. If meaningful metrics (time spent on a page, click-through rate, conversion of curiosity to sales) are available, randomized tests can be carried out to evaluate if the new technology is better than the current one. It is subtle what companies are looking to maximize in this scenario.

A longstanding and rich literature addresses the optimization of the so called *false discovery rate* (FDR) [75, 96]. Roughly speaking, it is the ratio of accepted technologies that are actually bad over the total number of implemented technologies. Then, an FDR minimization approach translates into accepting new technologies only when fairly confident that this would result an improvement. At a first glance this might seem like a sound goal, and fore some applications (e.g., testing medical treatments) this is indeed the case. However, for a company interested in maximizing profit it might not be. Consider this example. A first technology is proposed. It is slightly worse than the current one but the difference is negligible. Because of that, in order to detect that this is the case a large amount of time and resources have to be invested in testing. This has a negative effect for two reasons. First, testing might have a cost and spending a long time testing a technology that in the end is not even implemented is not what companies are looking for. Second, even if testing was free, spending a long time testing such a technology would result in a smaller profit due to missed opportunities. Assume, for example, that the second proposed technology is greatly superior to the current (and the first) one. A strategy that makes the first decision quickly and then moves on is also going to implement the second one quickly, because of the large difference in payoffs. Overall, even assuming that the company mistakenly implemented the fist technology, it is going to have a large increment in profit because of the implementation of the second one and the negligible negative effect of the first one. The first strategy, on the other hand, will stay stationary. This suggests that the right goal is making as many good decisions as possible in the shortest amount of time.

We formalize this idea by defining a notion of *reward per sample*, where the reward is the added value of accepted technologies. We discuss in depth this choice of performance measure and compare it with other natural alternatives. Two of the biggest technical issues with our choice is that our regret is not additive and running a policy does not return an unbiased estimates of its performance, where a policy is simply defined as a (possibly random) number of requested samples and a decision to accept or reject the tested technology. This prevents the use of standard bandit algorithms without major tweaking. Nevertheless, we design an algorithm with vanishing regret[2] even when applied to a countably infinite family of policies. Two key ideas are used in the analysis. The structure of the setting can be leveraged at an initial stage in order to reduce the size of the decision space. Then, a carefully limited use of oversampling can be employed in order to get a sufficient amount of unbiased estimates of the relevant policies. At a high level, we find the best finite number of actions hidden in an infinite pool of actions, then refine. A similar intuition will drive our dynamic pricing algorithms, whose introduction marks the beginning of the first chapter of this thesis.

[2]Being our regret normalized by the number of samples used, a vanishing regret corresponds to a sublinear regret in the bandit literature.

Chapter 1

Dynamic Pricing

Motivated by posted price auctions where buyers are grouped in an unknown number of latent types characterized by their private values for the good on sale, in this chapter we investigate revenue maximization in stochastic dynamic pricing when the distribution of buyers' private values is supported on an unknown set of points in $[0,1]$ of unknown cardinality K. This setting can be viewed as an instance of a stochastic K-armed bandit problem where the location of the arms (i.e., the K unknown valuations) must be learned as well.

1. In the distribution-free case, we show that our setting is just as hard as K-armed stochastic bandits: we prove that no algorithm can achieve a regret significantly better than $\sqrt{KT}$, (where T is the time horizon) and present an efficient algorithm matching this lower bound up to logarithmic factors.

2. In the distribution-dependent case, we show that for all $K > 2$ our setting is strictly harder than K-armed stochastic bandits by proving that it is impossible to obtain regret bounds that grow logarithmically in time or slower. On the other hand, when a lower bound $\gamma > 0$ on the smallest drop in the demand curve is known, we prove an upper bound on the regret of order $\big(1/\Delta + (\log\log T)/\gamma^2\big)\big(K\log T\big)$, where Δ is the gap between the revenue of the optimal valuation and that of the second-best valuation. This is a significant improvement on previously known regret bounds for discontinuous demand curves, that are at best of order $\big(K^{12}/\gamma^8\big)\sqrt{T}$.

3. When $K = 2$ in the distribution-dependent case, the hardness of our setting reduces to that of a stochastic 2-armed bandit: we prove that an upper bound of order $(\log T)/\Delta$ (up to $\log\log$ factors) on the regret can be achieved with no information on the demand curve.

4. Finally, we show a $\mathcal{O}(\sqrt{T})$ upper bound on the regret for the setting in which the buyers' decisions are nonstochastic, and the regret is measured with respect to the best between two fixed valuations one of which is known to the seller.

1.1 Introduction

In the online posted price auction problem, also known as dynamic pricing, an unlimited supply of identical goods is sold to a sequence of buyers. To each buyer in the sequence, the seller makes a take-it-or-leave-it offer for the good at a certain price (which we assume to belong to the unit interval $[0,1]$). The good is purchased if and only if the offered price is lower or equal to the buyer's private valuation (also assumed to be in $[0,1]$). At the end of the transaction, the seller's revenue is either zero (if the good is not sold) or equal to the offered price. The buyer's valuation is never observed. Indeed, the seller only learns a single bit for each auction, i.e., whether the good was sold or not at the chosen price. Similarly to previous works [52, 11, 10], we assume that the price offered to the t-th buyer in the sequence only depends on the past history of observed sales. In particular, we assume that buyers are indistinguishable, and provide no information to the seller other than their willingness to buy at the specified price. For this reason, the seller can post the price for the next buyer publicly, before the buyer shows up.

We evaluate the seller's performance in terms of regret, measuring the difference between the seller's revenue and the revenue achievable by consistently posting the optimal price. The regret in dynamic pricing was initially investigated by Kleinberg and Leighton [52] under various assumptions on the generation of the buyers' valuations. In the stochastic setting, in which valuations are drawn i.i.d. from a fixed and unknown distribution on $[0,1]$, they show that no algorithm can achieve a $o(\sqrt{T})$ regret and provide an algorithm achieving regret of order $C\sqrt{T \log T}$, where T is the number of buyers in the sequence and C only depends on the distribution of buyers' valuations. Their upper-bound holds under some assumptions on the demand curve, which is the function D mapping each price x to the probability $D(x) = \mathbb{P}(V \geq x)$ that the good is sold. Specifically, the revenue function $x \mapsto xD(x)$ is required to have a unique global maximum $x^\star \in (0,1)$ and be twice differentiable with a negative second derivative at $x^\star$. Without these assumptions, the authors prove a much higher lower bound of order $T^{2/3}$ on the regret. The algorithm achieving the $C\sqrt{T \log T}$ regret under the above assumptions on the demand curve is simple: it runs the UCB1 policy for stochastic bandits [4] on a discretized set of $K = (T/\log T)^{1/4}$ prices.

In this chapter, we study the stochastic setting of dynamic pricing under completely different assumptions on the demand curve. Namely, that the distribution of buyers' valuations is supported on an *unknown* set of *unknown* finite cardinality K. This models any setting in which buyers are grouped in an unknown number of latent types, characterized by their private values for the good on sale. In particular, this applies to regret minimization in sellers' repeated second-price auctions with a single relevant buyer. This scenario emerges naturally when a seller and a buyer interact repeatedly, and the valuation of the good depends on contextual information known only to the buyer. For instance, in online advertising each time a user lands on a publisher's website, an impression is put on sale to a set of relevant advertisers through an auction (note that whenever there is a single relevant advertiser for the impression, second-price auctions with reserve price are equivalent to posted price auctions). Now, typically, the advertiser's valuation for the impression depends on which segment the user belongs to, where the finite segmentation is based on private information not accessible to the publisher.

Note that our model is very different from assuming that the seller is restricted to offer prices from a *known* finite set of size K [78], which makes dynamic pricing a special case of K-armed stochastic bandits. In our model, the seller does not know the K buyers' valuations, not even their number! So, besides learning which valuation has the highest revenue, the seller must also learn the location of these values. This interplay between noisy search and bandit allocation is one of the main themes of our work.

In contrast with previous approaches, which typically assume parametric [16] or locally smooth [52] demand curves, our model with finitely many valuations is equivalent to assuming that the demand curve is piecewise constant with a finite number of discontinuities. Recently, den Boer and Keskin [29] designed an algorithm for piecewise continuous demand curves achieving an upper bound of order $C\sqrt{T}\log T$ in the piecewise constant case. However, up to constant factors, their hefty leading constant C is at least as big as the maximum between $K^{22}\gamma^{-16}c^{-2}$ and $K^{12}\gamma^{-8}c^{-18}$, where c is the minimum distance between valuations and both K and the smallest drop γ in the demand curve must be known in advance. Although their setting extends ours to certain piecewise *parametric* demand curves, we believe that discontinuities are the real source of additional hardness of this dynamic pricing model with respect to previously studied settings.

Our first result is a lower bound of order $\sqrt{KT}$ on the regret in the distribution-free case (where the regret is maximized over all possible demand curves), which holds even when the seller knows the number and position of buyers' private values in advance. This essentially establishes that our setting is at least as hard as a K-armed bandit problem. Although we build on the stochastic lower bound of Kleinberg and Leighton [52], our proof is not a simple adaptation of theirs. Indeed, we show that their proof breaks down when K is constant and T grows, which is exactly the regime we are interested in. Then, we present an efficient algorithm achieving a distribution-free upper bound on the regret of order $\sqrt{KT \log T}$ without any additional knowledge of the parameters of the problem.[1] The detailed version of our bound has a significantly better

[1]Throughout this chapter we assume that the time horizon T is known by the seller in advance. This assumption can be easily removed with a "doubling trick" (see, e.g., [23]), a standard technique for extending regret bounds to time sequences of unknown length.

dependence than den Boer and Keskin [29] on the smallest difference c between two adjacent valuations, and matches—up to logarithmic factors—the lower bound stated above.

In the distribution-dependent case, when the gap Δ between the revenue of the optimal valuation and that of the second-best valuation is constant, we prove the impossibility of obtaining regret bounds of order significantly better than $\sqrt{T}$ even when $K = 3$, thus showing that this setting is strictly harder than K-armed stochastic bandits. Motivated by this impossibility result, we investigate distribution-dependent bounds that rely on additional information about the demand curve. By combining suitable generalizations of UCB1 [4] and the "cautious search" strategy of Kleinberg and Leighton [52], we obtain an efficient algorithm achieving a regret of order at most $\bigl(1/\Delta + (\log\log T)/\gamma^2\bigr)\bigl(K \log T\bigr)$, where, as before, γ is the smallest drop in the demand curve. Since $(K/\Delta)\log T$ is the regret of K-armed stochastic bandits, this shows that the price of identifying each one of the K valuations is at most $(\log T)(\log\log T)/\gamma^2$, which corresponds (up to $\log\log$ factors) to the known upper bounds for noisy binary search [50]. We conclude the study of the distribution-dependent case by presenting an efficient algorithm with regret of order $(1/\Delta + \log\log T)\log T$ when the number of valuations is known to be at most two. Surprisingly, this bound is the same (up to $\log\log$ terms) as the best possible bound for two-armed stochastic bandits, achievable when not only the number, but also the locations of the valuations are known in advance. In order to prove this result we introduce a novel technique for estimating (up to a multiplicative constant) the expectation μ of any $[0,1]$-valued random variable with probability at least $1-\delta$, using at most $\mathcal{O}\bigl(\frac{1}{\mu}\ln\frac{1}{\delta}\bigr)$ samples, even if the expectation μ is *not* known in advance. We believe this technique may be valuable in its own right.

1.2 Further related works

The literature on dynamic pricing and online posted price auctions is vast. We address the reader to the excellent survey published by den Boer [30], providing a comprehensive picture of the state of the art until the end of 2014 —see also the tutorial slides by Slivkins and Zeevi [87] for a perspective more focused on computer science approaches. An important line of work in dynamic pricing considers a nonstochastic setting in which the sequence of the buyers' private values is deterministic and unknown, and the seller competes against the best fixed price. This model was pioneered by Kleinberg and Leighton [52], who proved a $\mathcal{O}(T^{2/3})$ upper bound (ignoring logarithmic factors) on the aforementioned notion of regret. Later works [11, 10] show simultaneous multiplicative and additive bounds on the regret when prices have range $[1,h]$. These bounds have the form $\varepsilon\, G_T^\star + \mathcal{O}\bigl((h \ln h)/\varepsilon^2\bigr)$ ignoring $\ln\ln h$ factors, where $G_T^\star$ is the total revenue of the optimal price $p^\star$. Recent improvements on these results are due to Bubeck et al. [21], who prove that the additive term can be made $\mathcal{O}(p^\star(\ln h)/\varepsilon^2)$, where the linear scaling is now with respect to the optimal price rather than the maximum price h. Other variants consider settings in which the number of copies of the item to sell is limited [1, 7, 8] or settings in which a returning buyer acts strategically in order to maximize his utility in future rounds [3, 31]

Finally, although in this work we focus on the seller's side, regret minimization approaches have been recently applied also on the buyer's side, for example in [61, 95].

1.3 Preliminaries and definitions

We assume all valuations V_t belong to a fixed and unknown finite set $\mathcal{V} = \{v_1, \dots, v_K\} \subset [0,1]$, with $0 = v_0 \le v_1 < \cdots < v_K \le v_{K+1} = 1$. Unless otherwise specified, the sequence $V_1, V_2, \dots$ is assumed to be sampled i.i.d. from a fixed and unknown distribution on $\{v_1, \dots, v_K\}$. Let $p_i = \mathbb{P}(V_1 = v_i)$ and assume (without loss of generality) that $p_i > 0$ for all $i \in \{1, \dots, K\}$. An instance of the posted price problem is then fully specified by the pairs $(v_1, p_1), \dots, (v_K, p_K)$. We assume auctions are implemented according to the following online protocol: for each round $t \in \{1, 2, \dots\}$

1. the seller posts a price $X_t \in [0,1]$

2. buyer's valuation V_t, hidden from the seller, is drawn from $\mathcal{V}$ according to $\{p_1, \dots, p_K\}$

3. the seller observes $\mathbb{I}\{V_t \geq X_t\} \in \{0,1\}$ and computes the revenue $r_t(X_t) = X_t \mathbb{I}\{V_t \geq X_t\}$

Note that the expected revenue $\mathbb{E}[r_t(x)] = \mathbb{E}\big[x\,\mathbb{I}\{V_t \geq x\}\big]$ is equal to $x\,D(x)$, where

$$D(x) = \mathbb{P}(V_1 \geq x) = \sum_{k\colon v_k \geq x} p_k \tag{1.1}$$

is the *demand curve.* Hence the price maximizing the expected revenue $\mathbb{E}[r_t(x)]$ belongs to the set of valuations $\{v_1, \dots, v_K\}$ and we denote one of the possible optimal valuations by $v^\star = v_{i^\star}$. We define the suboptimality gap of v_j with respect to $v^\star$ by $\Delta_j = \mathbb{E}\big[r_1(v^\star) - r_1(v_j)\big]$. The goal of the seller is to minimize the *regret*

$$R_T = \max_{x\in[0,1]} \mathbb{E}\left[\sum_{t=1}^{T} r_t(x) - \sum_{t=1}^{T} r_t(X_t)\right] = \mathbb{E}\left[\sum_{t=1}^{T} r_t(v^\star) - r_t(X_t)\right]$$

where the expectation is understood with respect to any randomness in the generation of $V_1, \dots, V_T$ and $X_1, \dots, X_T$. Formally, a *deterministic seller* is a sequence of functions $X_1, X_2, \dots$ where each function $X_t = f_t(X_1, Z_1, \dots, X_{t-1}, Z_{t-1})$ is the price posted at time t, the random variable Z_s is the binary feedback $\mathbb{I}\{V_s \geq X_s\}$ received by the seller in at time s, and $f_t\colon \big([0,1] \times \{0,1\}\big)^{t-1} \to [0,1]$ is an arbitrary function. A *randomized seller* is a probability distribution over deterministic sellers.

1.4 Lower bounds

In this section we show some important similarities and differences between dynamic pricing with K valuations and the K-armed bandit problem. First, we state that in the distribution-free case the former is at least as difficult as the latter. More precisely, if $T \geq K^3$, no algorithm can have regret better than $\sqrt{KT}$ on dynamic pricing with K valuations. The proof of the following theorem is deferred to Section 1.8.1.

Theorem 1.1. *For any number of valuations $K \geq 3$ and all time horizons $T \geq K^3$ there exist K pairs $(v_1, p_1), \dots, (v_K, p_K)$ such that the expected regret of any pricing strategy satisfies $R_T = \Omega\big(\sqrt{KT}\big)$.*

Next, we show that in the distribution-dependent case, dynamic pricing is strictly harder than multiarmed bandits. More precisely, even if the suboptimality gap Δ is constant and K is small, no dynamic pricing algorithm can have regret better than $\sqrt{T}$, whereas the distribution-dependent regret of multiarmed bandits is $\mathcal{O}(\log T)$.

Theorem 1.2. *If for some constant $c^\star > 0$ a seller algorithm has regret smaller than $c^\star\sqrt{T}$ on any instance of the stochastic dynamic pricing problem with at most three valuations, then there exists an instance with $\Delta = \Theta(1)$ on which the algorithm suffers regret $\Omega(\sqrt{T})$.*

Proof. We consider two instances. The first has $\Delta = \frac{1}{4}$ and the second has $\Delta = \mathcal{O}(1/\sqrt{T})$. We prove that if the algorithm has regret $\mathcal{O}(\sqrt{T})$ on both instances, then it must have regret $\Omega(\sqrt{T})$ on the first instance. The two instances are defined as follows.

Instance 1

$v_1^{(1)} = 0$	$D^{(1)}(0) = 1$	$r^{(1)}(0) = 0$
$v_2^{(1)} = \frac{1}{2}$	$D^{(1)}\big(\frac{1}{2}\big) = \frac{1}{2}$	$r^{(1)}\big(\frac{1}{2}\big) = \frac{1}{4}$

Instance 2

$v_1^{(2)} = 0$	$D^{(2)}(0) = 1$	$r^{(2)}(0) = 0$
$v_2^{(2)} = \frac{1-\eta}{2}$	$D^{(2)}\big(\frac{1-\eta}{2}\big) = \frac{1}{2} + \eta$	$r^{(2)}\big(\frac{1-\eta}{2}\big) = \frac{1+\eta-2\eta^2}{4}$
$v_3^{(2)} = \frac{1}{2}$	$D^{(2)}\big(\frac{1}{2}\big) = \frac{1}{2}$	$r^{(2)}\big(\frac{1}{2}\big) = \frac{1}{4}$

In Instance 1 the optimal price is $v_2^{(1)} = \frac{1}{2}$ with revenue $\frac{1}{4}$. In Instance 2 the optimal price is $v_2^{(2)} = \frac{1-\eta}{2}$ with revenue $\frac{1+\eta-2\eta^2}{4} \geq \frac{1}{4} + \frac{\eta}{8}$ for $\eta \leq \frac{1}{4}$. Without loss of generality, we can assume that the seller algorithm only posts prices in the set $\big\{0, \frac{1-\eta}{2}, \frac{1}{2}\big\}$. Let $N_\eta(t)$ be the number of times that the price $\frac{1-\eta}{2}$ is posted and let $\nu_t^{(i)}$ be the law of observed rewards up to time t in Instance $i \in \{1,2\}$. Since prices 0 and $\frac{1}{2}$ are uninformative (because demand and revenue do no change across the two instances), it follows from standard calculations that the KL divergence between $\nu_t^{(1)}$ and $\nu_t^{(2)}$ is upper bounded by the KL between two Bernoulli of parameter $\frac{1}{2}$ and $\frac{1}{2}+\eta$ times the expected number of times v_2 is chosen under Instance 1,

$$\mathrm{KL}\big(\nu_t^{(1)} \,\|\, \nu_t^{(2)}\big) \le \mathrm{KL}\left(\frac{1}{2} \,\middle\|\, \frac{1}{2}+\eta\right) \mathbb{E}_1\big[N_\eta(t)\big] \le 4\eta^2\, \mathbb{E}_1\big[N_\eta(t)\big] \qquad \text{if } \eta \le \frac{1}{4}$$

where $\mathbb{E}_1$ denotes expectation under Instance 1. Let $R_T^{(i)}$ be the regret under Instance $i \in \{1,2\}$. Since $r^{(1)}\big(\frac{1-\eta}{2}\big) = \frac{1-\eta}{2} D^{(1)}\big(\frac{1-\eta}{2}\big) = \frac{1-\eta}{4}$, we have $R_T^{(1)} \ge \frac{\eta}{4}\mathbb{E}_1\big[N_\eta(T)\big]$. Using the assumption that the seller's algorithm has a regret smaller than $c^\star\sqrt{T}$, and adapting an argument of Bubeck et al. [20, Proof of Theorem 5], we can write

$$\frac{\eta}{4}\frac{T}{4}\exp\Big(-4\eta^2\mathbb{E}_1\big[N_\eta(T)\big]\Big) \le \max\Big\{R_T^{(1)}, R_T^{(2)}\Big\} \le c^\star\sqrt{T}\ .$$

Hence, for $\eta = \frac{32c^\star}{\sqrt{T}}$, it must hold that $\mathbb{E}_1\big[N_\eta(t)\big] \ge \frac{\ln 2}{4\eta^2}$, which implies that $R_T^{(1)} \ge \frac{\ln 2}{512c^\star}\sqrt{T}$. $\square$

Theorem 1.2 can be extended to the case when K is known to the seller. This can be done by adding an extra valuation $v_3^{(1)} > v_2^{(1)}$ to Instance 1 which has either vanishing probability p_3 or vanishing distance $v_3^{(1)} - v_2^{(1)}$ from $v_2^{(1)}$. (In the latter case the value of $v_3^{(1)}$ depends on the algorithms.) In both cases the seller algorithm is unlikely to detect the presence of this extra valuation, and a slight modified proof of Theorem 1.2 can be applied.

This lower bound shows that $\sqrt{T}$ is best possible in the distribution-dependent case even when K is small and Δ is a constant. In Section 1.6 we show how regret bounds can be substantially better than $\sqrt{T}$ when the learner knows the value of the smallest drop in the demand curve.

1.5 Distribution-free bounds

In this section we focus on distribution-free bounds, i.e., bounds that do not depend on the demand curve. The regret bound we prove exceeds the theoretical lower bound stated in Section 1.4 by a constant term depending only on the distance between adjacent valuations.

Our Algorithm 1 works in two phases: a search phase and a bandit phase. In the search phase a binary search for all "relevant" valuations is performed. By the end of this phase, a tight estimate of all such valuations is determined with high probability. During the bandit phase a stochastic bandit algorithm is run on the estimated valuations. As it turns out, this simple scheme is enough to ensure an optimal $\sqrt{KT}$ convergence up to an additive constant independent of the distribution of buyer's valuations. Notably, the algorithm *does not* need to know K in advance. We call *macrostep* a block of consecutive rounds in which the same price is offered consistently. For each price x we denote by $\overline{D}(x)$ the fraction of accepted offers of x during the last macrostep in which x was offered. At the beginning of the search phase, our algorithm receives as input the time horizon T and a confidence parameter δ. The algorithm then proceeds in macrosteps of length $\big\lceil 8\sqrt{T/k_m}\ln\delta^{-1}\big\rceil$, where k_m is the total number of valuations discovered so far. The goal of the search phase is to approximately locate all *relevant* valuations, that is valuations v_i whose associated probability p_i is at least $\sqrt[4]{K/T}$.

Initially, all relevant valuations belong to $[a_1, b_1] = [0,1]$. The search proceeds as long as there is at least an interval i containing relevant valuations with length larger than $T^{-1/2}$ (line 2). When such an interval i is selected at line 3, a macrostep of binary search is performed and the midpoint price x_m of $[a_i, b_i]$ is offered for $\big\lceil 8\sqrt{T/k_m}\ln\delta^{-1}\big\rceil$ rounds (line 4), thus obtaining an estimate of its demand. If the difference in demands (line 5) is smaller than $(k_m/T)^{-1/4}/2$ no new relevant valuation is detected. Before eliminating the lower half of the interval (line 7), a test designed to detect and remove *fake arms* is performed (line 6). We call fake arm an interval containing no relevant valuations. Fake arms might be inadvertently allocated when intervals are too wide. In that case, the comparison between two distant points may reveal a large difference in demands due to the presence of several nonrelevant valuations in between. If that happens, the fake arm is removed when the interval becomes small enough (line 8). When no significant difference is

Algorithm 1:

```
Input: $T \in \mathbb{N}$, $\delta \in (0,1)$.
Initialization: $\mathcal{K}_1 \leftarrow \{1\}$, $k_1 \leftarrow 1$, $a_1 \leftarrow 0$, $b_1 \leftarrow 1$, $a_0 \leftarrow 0$, $\overline{D}(0) \leftarrow 0$.
1  for $m = 1, 2, \dots$ do                                                      // search phase
2      if $\{j \in \mathcal{K}_m \mid b_j - a_j > T^{-1/2}\} \neq \varnothing$ then
3          pick $i_m = \min\{j \in \mathcal{K}_m \mid b_j - a_j > T^{-1/2}\}$;
4          offer price $x_m = (a_{i_m} + b_{i_m})/2$ for $\lceil 8\sqrt{T/k_m} \ln \delta^{-1} \rceil$ rounds;
5          if $\overline{D}(a_{i_m}) - \overline{D}(x_m) < (k_m/T)^{1/4}/2$ then                    // undershooting
6              if $\overline{D}(x_m) - \overline{D}(b_{i_m}) \geq (k_m/T)^{1/4}/2$ then             // check for fake arms
7                  update $a_{i_m} \leftarrow x_m$, $\mathcal{K}_{m+1} \leftarrow \mathcal{K}_m$ and $k_{m+1} \leftarrow k_m$;
8              else update $\mathcal{K}_{m+1} \leftarrow \mathcal{K}_m \setminus \{i_m\}$ and $k_{m+1} \leftarrow k_m$;
9          else if $\overline{D}(a_{i_m}) - \overline{D}(x_m) \geq (k_m/T)^{1/4}/2$ then                // overshooting
10             if $\operatorname{sign}(a_i - x_m)\big(\overline{D}(a_i) - \overline{D}(x_m)\big) \geq (k_m/T)^{1/4}/2$ for all $i$ then    // new arms
11                 set $a_{k_m+1} \leftarrow x_m$, $b_{k_m+1} \leftarrow b_{i_m}$, $\mathcal{K}_{m+1} \leftarrow \mathcal{K}_m \cup \{k_m + 1\}$ and $k_{m+1} \leftarrow k_m + 1$;
12             update $b_{i_m} \leftarrow x_m$, $\mathcal{K}_{m+1} \leftarrow \mathcal{K}_m$ and $k_{m+1} \leftarrow k_m$;
13     else denote the last macrostep by $M$ and break;
14 run the UCB1 algorithm on the set of prices $\{a_j\}_{j \in \mathcal{K}_M}$;                   // bandit phase
```

detected between the demands, all relevant valuations in $[a_i, b_i]$ remain in $[x_m, b_i]$ with high probability after the update. If, on the other hand, a difference between demands is detected (line 9), two things happen. First, a test is performed to detect possible new relevant valuations (line 10). If a new relevant valuation is spotted, a new interval $[x_m, b_i]$ is allocated. Second, the upper half of the interval $[a_i, b_i]$ is removed. If $[a_i, b_i]$ is split into $[a_i, x_m]$ and $[x_m, b_i]$, all relevant valuations are split between the two intervals. If $[a_i, b_i]$ is simply updated as $[a_i, x_m]$—since no significant difference was detected between the demands at x_m and b_i—all relevant valuations in $[a_i, b_i]$ remain in $[a_i, x_m]$ with high probability.

When all intervals become smaller than $T^{-1/2}$ (line 13), the search phase ends and all intervals $[a_i, b_i]$ are returned. At this point each relevant valuation is contained in one of the intervals with high probability. Therefore the algorithm has now access to $T^{-1/2}$-close approximations of all of them, and the bandit phase begins. In the bandit phase, the algorithm UCB1 [4] is run on the set of left endpoints of the intervals (line 14).

Theorem 1.3. *If Algorithm 1 is run on an unknown number K of pairs $(v_1, p_1), \dots, (v_K, p_K)$ with input parameter $\delta = T^{-2}$, then its regret satisfies*

$$R_T = \widetilde{\mathcal{O}}\left(\sqrt{KT}\right) + V(V+1) \qquad \textit{where} \quad V = \max_{i \in \{1,\dots,K\}} \frac{v_k^4}{(v_i - v_{i-1})^5} \ .$$

We actually prove a slightly improved bound, in which the constant $V(V+1)$ is replaced by the smaller term $K(v_K^4/v_1^4)\big(1 + (v_K^4/c^4)\big)$, where $c = \min_{i \in \{2,\dots,K\}}\{v_i - v_{i-1}\}$. To give a frame of reference, previously known upper bounds for discontinuous demand curves [29] are at best of order $\big(K^{20}/c^{18}\big)\sqrt{T}$, where v_1 is assumed to be bounded away from zero and K needs to be known in advance.

Proof. We begin by proving that at any time time during the search phase, all intervals $[a_i, b_i]$ satisfy $D(b_i) - D(a_i) \geq T^{-1/4}$ with high probability and with the same probability all valuations v_j not belonging to any of these intervals satisfy $p_j < (K/T)^{1/4}$. For any given price $x \in [0,1]$ offered during the search phase, Hoeffding's inequality implies $\big|\overline{D}(x) - D(x)\big| \leq (|\mathcal{K}_m|/T)^{1/4}/4$ with probability at least $1 - 2\delta$. Therefore, if $D(x) - D(y) \geq (|\mathcal{K}_m|/T)^{1/4}$, then $\overline{D}(x) - \overline{D}(y) \geq (|\mathcal{K}_m|/T)^{1/4}/2$ with probability at least $1 - 2\delta$. Moreover, if $D(x) = D(y)$, then $\overline{D}(x) - \overline{D}(y) < (|\mathcal{K}_m|/T)^{1/4}/2$ with probability at least $1 - 4\delta$. Since at each macrostep the algorithm performs at most $K + 1$ comparisons between $\overline{D}(x)$ and $\overline{D}(y)$ for pairs of points x, y (lines 5, 6, 10), the probability that, for at least one of these comparisons, we have

$$\left(\overline{D}(x) - \overline{D}(y) < \sqrt[4]{\tfrac{k_m}{16\,T}} \;\wedge\; D(x) - D(y) \geq \sqrt[4]{\tfrac{k_m}{T}}\right) \text{ or } \left(\overline{D}(x) - \overline{D}(y) \geq \sqrt[4]{\tfrac{k_m}{16\,T}} \;\wedge\; D(x) = D(y)\right) \tag{1.2}$$

is at most $4(K+1)\delta$. Thus the probability that the event (1.2) occurs for at least one comparison in at least one macrostep is at most $4(K+1)M\delta$, where $M \leq \sqrt{KT}$. This proves the initial claim. By paying an additional $4(K+1)M\delta T = \mathcal{O}\big(K\sqrt{K/T}\big)$ we can therefore assume that event (1.2) never occurs. In this case at most K binary searches are performed and —ignoring constants and logarithmic factors— the regret increases by at most $\sum_{k=1}^{K}\sqrt{T/k} \leq \sqrt{T}\int_0^K x^{-1/2}\mathrm{d}x = 2\sqrt{KT}$. We prove now that if $v_K \notin \bigcup_{j\in\mathcal{K}_M}[a_j,b_j]$ (which implies $p_K < \sqrt[4]{K/T}$), then it is suboptimal. In order for v_K to be optimal, it would have to have at least a revenue higher than v_1. Thus

$$v_K p_K \geq v_1 \implies \sqrt[4]{K/T} > p_K \geq \frac{v_1}{v_K}$$

which can only happen if $T < K(v_K/v_1)^4$. By paying an additional $K(v_K/v_1)^4$ term in the regret we can therefore assume that v_K is suboptimal. We show now that all other valuations not belonging to $\bigcup_{j\in\mathcal{K}_M}[a_j,b_j]$ are also suboptimal. Take any valuation $v_j \notin \bigcup_{i\in\mathcal{K}_M}[a_i,b_i]$ (which again, implies $p_j < \sqrt[4]{K/T}$) strictly smaller than v_K. In order for v_j to be optimal, it has to at least be better than v_1 and v_j+1. If v_j is better than v_1

$$v_j\sum_{k=j}^{K} p_k \geq v_1 \implies p_{j+1} \geq \frac{v_1}{v_j} - p_j - \sum_{k=j+2}^{K} p_k. \tag{1.3}$$

If v_j is better than v_{j+1}

$$v_j\sum_{k=j}^{K} p_k \geq v_{j+1}\sum_{k=j+1}^{K} p_k \implies p_j \geq \left(\frac{v_{j+1}}{v_j} - 1\right)\sum_{k=j+1}^{K} p_k = \left(\frac{v_{j+1}}{v_j} - 1\right)\left(p_{j+1} + \sum_{k=j+2}^{K} p_k\right)$$

and lower bounding p_{j+1} as in (1.3) gives

$$p_j \geq \left(\frac{v_{j+1}}{v_j} - 1\right)\left(\frac{v_1}{v_j} - p_j\right) \implies p_j \geq \frac{v_1}{v_{j+1}}\,\frac{v_{j+1}-v_j}{v_j} \geq \frac{v_1 c}{v_K^2}$$

where $c = \min_{i\in\{2,\dots,K\}}\{v_i - v_{i-1}\}$. Being $p_j < \sqrt[4]{K/T}$ this can only happen if $T < Kv_K^8/(v_1c)^4$. Thus we can assume v_j is suboptimal by paying at most an extra $Kv_K^8/(v_1c)^4$ term in the regret. This proves that $v^\star \in \bigcup_{j\in\mathcal{K}_M}[a_j,b_j]$. Being $b_j - a_j < T^{-1/2}$, offering a_j rather than any $x \in [a_j,b_j]$ results in an regret increase of at most $\sqrt{T}$. Finally, running the UCB1 algorithm [4] for standard stochastic bandits adds another $\widetilde{\mathcal{O}}(\sqrt{k_m T})$ term to the regret, where again $k_m \leq K$. □

We now discuss the role that c and v_1 play in the dynamic pricing problem. Assume that $p_{i^\star} < \sqrt[4]{K/T}$ but there exist valuations $v_j > v^\star$ with $p_j \geq \sqrt[4]{K/T}$, and let v_k be the smallest of such valuations. Arguing as in the proof of Theorem 1.3, one can prove that $d_k = v_k - v^\star$ must satisfy

$$d_k \leq \frac{v_K^2}{v_1}K\sqrt[4]{\frac{K}{T}}\;.$$

This means that in principle the optimum valuation $v^\star$ could be hiding in any of the intervals $[v_i - d_i, v_i - c)$, where v_i are all valuations with probabilities $p_i \geq \sqrt[4]{K/T}$. Since these intervals become bigger and bigger as c approaches zero, this behavior foils the attempt of identifying the finite support of the problem instance. The smallest valuation v_1 is also a natural parameter of the problem for an entirely different reason. Indeed v_1 is not just a valuation, it is the only valuation which is also its own revenue. Assume for example that $v_1 = 0$

```
Algorithm 2:
Input: Time horizon $T \in \mathbb{N}$.
Initialization: set $\kappa_0 \leftarrow 1$, $a_1 \leftarrow 0$, $b_1 \leftarrow 1$, $n_1 \leftarrow 1$, $\varepsilon_1 \leftarrow 1/2$, $D_1 \leftarrow 1$.
for $t = 1$ to $T$ do
    set $\kappa_t \leftarrow \kappa_{t-1}$;
    compute $i_t \leftarrow \arg\max_{i \le \kappa_t} b_i D_i$;                         // greedy pick
    if $b_{i_t} - a_{i_t} \le 1/T$ then post $a_{i_t}$;        // if $[a_{i_t}, b_{i_t}]$ becomes tiny, play $a_{i_t}$ for good
    else
        post $X_t = a_{i_t} + n_{i_t}\varepsilon_{i_t}$ and get feedback $X_t D(X_t)$;
        if $D(X_t) = D_{i_t}$ then          // increase prices until surpassing the closest $v_j$
            if $X_t + \varepsilon_{i_t} < b_{i_t}$ then update $n_{i_t} \leftarrow n_{i_t} + 1$;
            else update $a_{i_t} \leftarrow X_t$, $n_{i_t} \leftarrow 1$, $\varepsilon_{i_t} \leftarrow \varepsilon_{i_t}^2$;              // shrink the interval
        else
            if $D(X_t) \notin \{D_1, \dots, D_{\kappa_t}, 0\}$ then                // a new valuation is found
                set $\kappa_t \leftarrow \kappa_{t-1} + 1$, $a_{\kappa_t} \leftarrow X_t$, $b_{\kappa_t} \leftarrow b_{i_t}$, $n_{\kappa_t} \leftarrow 1$, $\varepsilon_{\kappa_t} \leftarrow \varepsilon_{i_t}$, $D_{\kappa_t} \leftarrow D(X_t)$;
            update $a_{i_t} \leftarrow X_t - \varepsilon_{i_t}$, $b_{i_t} \leftarrow X_t$, $n_{i_t} \leftarrow 1$, $\varepsilon_{i_t} \leftarrow \varepsilon_{i_t}^2$;        // shrink the interval
```

(which makes it always suboptimal). Even if this piece of information is known by the seller, and the problem is reduced to $\{v_2, \dots, v_K\} \subset (0, 1]$, the reduced problem becomes harder as the "weights" $\{p_2, \dots, p_K\}$ do not sum to 1 anymore. The worst case happens when p_1 is close to 1. In this case a considerable amount of samples is needed just to locate any of the remaining valuations, let alone the optimal one, in an online fashion, while accruing regret at each round.

1.6 Distribution-dependent bounds

In this section we focus on distribution-dependent bounds, i.e., bounds that are parameterised in terms of the demand curve. Our algorithm ignores the number of valuations, but is given a lower bound γ on the smallest probability $p_{\min}$ of a valuation (i.e., the smallest drop in the demand) —note that $\gamma \le p_{\min}$ implies $K \le 1/\gamma$, so we also have an upper bound on the number of valuations. The regret bound we prove exceeds the distribution-dependent regret $(K \ln T)/\Delta$ of standard stochastic bandits by a term of order $K(\ln T)(\ln\ln T)/\gamma^2$. On the other hand, if the number K of valuations (counting only those which are at least T^{-1} apart) is exactly known, it is easy to prove an excess regret bound of order $K((\ln T)/p_{\min})^2$ even when $p_{\min}$ (or a lower bound on it) is unknown: The algorithm performs $\mathcal{O}(\ln T)$ binary search steps for each one of the K valuations, repeating each step $\mathcal{O}((\ln T)/\gamma^2)$ times and using a value of γ that decreases geometrically until all K valuations are found. A similar argument gives the same regret bound in the case when K not known exactly, but $\gamma \le p_{\min}$ and $c \le \min_k(p_k - p_{k-1})$ are both known.

In order to introduce in a clear and concise manner the ideas used to prove our main result, we begin by considering an easier setting in which the feedback is provided by an oracle returning the value of the demand curve $D(X_t)$ at the posted price X_t. This is equivalent to assuming that the feedback is the expectation $\mathbb{E}\big[r_t(X_t) \mid V_1, \dots, V_{t-1}\big] = X_t D(X_t)$ rather than the random variable $r_t(X_t)$. This simplified setting allows us to focus on the search of the valuations points, abstracting from the problem of estimating the demand curve. We define a seller algorithm that extends the "cautious search" strategy for a single unknown valuation ([52], see Algorithm 4 in Section 1.8.2) to an unknown number of unknown valuations.

Our algorithm (Algorithm 2) initially looks for a single valuation v_1, and then allocates searches for new valuations incrementally. Whenever a new value of the demand curve is observed, providing evidence for the existence of a i-th previously unseen valuation, an interval $[a_i, b_i]$ (which we associate with a bandit arm) and a step size ε_i are allocated. The interval $[a_i, b_i]$ estimates the smallest valuation v_i contained in it. By construction of the algorithm, v_i is never removed from $[a_i, b_i]$ when the interval shrinks. This implies that the more $[a_i, b_i]$ shrinks, the closer $b_i D(a_i)$ gets to the true revenue $v_i D(v_i)$.

The algorithm works by performing cautious searches within each interval. At the beginning, all valuations belong to $[a_1, b_1] = [0, 1]$. Whenever an interval is selected (line 3), a step of cautious search is performed (lines 6–13). During a cautious search in $[a_i, b_i]$ with step size ε_i, the sequence of values $X_t = a_i + k\varepsilon_i$ for $k \in \{1, 2, \dots\}$ is posted until a change is spotted in the demand or X_t gets within ε_i of b_i. If the latter happens before a change in the demand is discovered (line 8), the interval shrinks to $[X_t, b_{i_t}]$ and the step size is refined (line 9). Note that the shrunken interval contains all valuations that were in $[a_i, b_i]$ because the demand did not change. If a change in the demand is spotted (line 10), then the interval shrinks to $[X_t - \varepsilon_i, X_t]$ and the step size is reduced (line 13). If the new demand value matches the value of $D(b_i)$ the shrunken interval contains again all valuations that were in $[a_i, b_i]$. If the new demand value does not belong to a known interval (line 11), then a new interval $[X_t, b_i]$ is allocated (line 12). This process continues until the length of the feasible interval $[a_j, b_j]$ of the arm j with the highest $b_j D_j$ is less than $1/T$. Then the seller offers the same price a_j for all remaining rounds. As time goes by, the number κ of discovered valuations grows until possibly reaching the actual number of valuations K. Simultaneously, each estimate $b_i D_i$ converges to the revenue of the smallest valuation in the interval. After enough rounds, picking the interval i with the highest $b_i D_i$ becomes equivalent to choosing a $1/T$-approximation of an optimal valuation. Without loss of generality, in the analysis of the algorithm, we assume all valuations $v_1, \dots, v_K$ are at least $1/T$ apart. Let i_s be the index of the arm chosen at time s (line 3). For any $k = \{1, \dots, K\}$, let $\mathcal{T}_k \in \{t \le T \mid v_k \in [a_{i_t}, b_{i_t}]\}$. The next lemma states that the steps performed by Algorithm 2 in all the intervals that ever contained v_k are those that a cautious search would have performed if run on the single evaluation v_k.

Lemma 1.1. *Suppose Algorithm 2 is run on K valuations $v_1, \dots, v_K$. Pick $k \in \{1, \dots, K\}$ and $n \in \{1, \dots, |\mathcal{T}_k|\}$. Let $[0, 1] \equiv I_1 \supseteq \dots \supseteq I_n \equiv [a'_n, b'_n]$ be the sequence of the first n intervals computed by n steps of a cautious search for the single valuation v_k with initial interval $[0, 1]$. Then $a'_n \le a_{i_t}$ and $b'_n = b_{i_t}$, where t is the n-th smallest value in $\mathcal{T}_k$. Moreover, the price X_t offered by Algorithm 2 at time t is equal to the n-th price offered by the cautious search for the single valuation v_k.*

Proof. Fix a valuation v_k. Let A be Algorithm 2 and C be the cautious search for v_k. The proof is by induction on n. Since A and C both start with interval $[0, 1]$ and price $1/2$ the statement holds for $n = 1$. Now let t be the $(n + 1)$-st smallest value in $\mathcal{T}_k$ and let s be the largest value in $\mathcal{T}_k$ that is smaller than t. Let $I_n \equiv [a'_n, b'_n]$ be the n-th interval computed by C. By induction, $a'_n \le a_{i_s}$, $b'_n = b_{i_s}$, and X_s is offered by both A and C. The only interesting case to discuss is when the test at line 7 is false. There are two subcases: if the test at line 11 is false, then it must be $X_s > v_k$. In this case C overshoots and the interval is updated exactly in the same way by C and A (see line 13). If the test at line 11 is true, then it must be $v_i < X_s \le v_k$. This is not an overshoot for C, so $I_{n+1} \equiv I_n$. A, however, creates a new interval $[a, b]$ —containing v_k— with $a = X_s$, $b = b_{i_s}$, and unchanged step size ε_{i_s}. The next time t this new interval is selected, the price X_t offered by A is the same as the price offered by C because the step size did not change. □

Theorem 1.4. *If Algorithm 2 is run on an unknown number K of pairs $(v_1, p_1), \dots, (v_K, p_K)$, then its regret satisfies $R_T \le K(3 \ln \ln T + 10)$.*

Proof. Intervals are indexed in their order of creation (so that interval 1 is $[0, 1]$), and the i-th interval is identified with bandit arm i. Note that, at any point during the execution of the algorithm, each valuation belongs to some interval. Any interval is created with at least one valuation in it, and shrinks until it only contains the smallest valuation v_j among those that initially belonged to it. Let κ_T be the number of intervals created after T rounds. For $i \in \{1, \dots, \kappa_T\}$, denote by $\mu(i)$ the index $j \in \{1, \dots, K\}$ of the smallest $v_j \in [a_i, b_i]$. Now fix any k such that $k = \mu(j)$ (i.e., v_k is the smallest value of the j-th interval) for some $j \in \{1, \dots, \kappa_T\}$. Note that $j = i_{t_k}$ for some $t_k \in \{1, \dots, T\}$ because $k = \mu(j)$ implies that when interval j is created v_k is its smallest valuation. Hence the last selected interval containing v_k must be j. Let $T_k = |\mathcal{T}_k|$ and $t_k = \max \mathcal{T}_k$. Lemma 1.1 implies that at time t_k the overall number of cautious steps made for v_k is T_k, Lemma 1.5 implies $b_j - a_j \le 2/T_k$ at time t_k. Now note that $D_j = D(v_k)$ because $k = \mu(j)$. Since $v^\star$ belongs to some $[a_{i^\star}, b_{i^\star}]$, and using $D_{i^\star} = D(a_{i^\star})$, at time t_k we have $v^\star D(v^\star) \le b_{i^\star} D_{i^\star} \le b_j D_j = b_j D(v_k) \le \big(v_k + (b_j - a_j)\big) D(v_k)$. Then the above implies $T_k \le 2D(v_k)/\Delta_k$ where $\Delta_k = v^\star D(v^\star) - v_k D(v_k)$.

Algorithm 3:

```
Input: Time horizon $T \in \mathbb{N}$, confidence parameter $\delta \in (0,1)$.
Initialization: set $\kappa_0 = 1$, $a_1 \leftarrow 0$, $b_1 \leftarrow 1$, $n_1 \leftarrow 1$, $\varepsilon_1 \leftarrow 1/2$, $\overline{D}(a_1) = 1$.
1  for $m = 1$ to $M_\gamma$ do
2      set $\kappa_m \leftarrow \kappa_{m-1}$;
3      compute $i_m \leftarrow \arg\max_{i \le \kappa_m} b_i U_i$;                          // greedy pick
4      if $b_{i_m} - a_{i_m} \le 1/T$ then post $a_{i_m}$;     // if $[a_{i_m}, b_{i_m}]$ gets tiny, play $a_{i_m}$ for good
5      else
6          post $X_m = a_{i_m} + n_{i_m}\varepsilon_{i_m}$ for $\lceil 8\ln(\delta^{-1})/\gamma^2 \rceil$ rounds and compute $\overline{D}(X_m)$;
7          if $\overline{D}(a_{i_m}) - \overline{D}(X_m) < \gamma/2$ then     // up prices until surpassing the closest $v_j$
8              if $X_m + \varepsilon_{i_m} < b_{i_m}$ then update $n_{i_m} \leftarrow n_{i_m} + 1$;
9              else update $a_{i_m} \leftarrow X_m$, $n_{i_m} \leftarrow 0$, $\varepsilon_{i_m} \leftarrow \varepsilon_{i_m}^2$;          // shrink the interval
10         else (denoting $a_0 = \overline{D}(0) = 0$)
11             if $\forall i \ne i_m$, $\mathrm{sign}(a_i - X_m)\big(\overline{D}(a_i) - \overline{D}(X_m)\big) \ge \gamma/2$ then          // new valuation
12                 $\kappa_m \leftarrow \kappa_{m-1} + 1$, $a_{\kappa_m} \leftarrow X_m$, $b_{\kappa_m} \leftarrow b_{i_m}$, $n_{\kappa_m} \leftarrow 1$, $\varepsilon_{\kappa_m} \leftarrow \varepsilon_{i_m}$;
13             update $a_{i_m} \leftarrow X_m - \varepsilon_{i_m}$, $b_{i_m} \leftarrow X_m$, $n_{i_m} \leftarrow 0$, $\varepsilon_{i_m} \leftarrow \varepsilon_{i_m}^2$;     // shrink interval
```

Lemma 1.4 and Lemma 1.1 also imply

$$\sum_{t \in \mathcal{T}_k} \big(r_t(v_k) - r_t(X_t)\big) \le 3 \ln\ln T_k + 8 \ . \tag{1.4}$$

Noting that $\{1, \dots, T\} \subseteq \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_K$, we may write

$$\begin{aligned}
R_T &= \sum_{t=1}^{T} \Big(r_t(v^\star) - r_t(X_t)\Big) \le \sum_{k=1}^{K} \sum_{t \in \mathcal{T}_k} \Big(r_t(v^\star) - r_t(X_t)\Big) \\
&\le \sum_{k=1}^{K} \Big(T_k v^\star D(v^\star) - \big(T_k v_k D(v_k) - (3\ln\ln T_k + 8)\big)\Big) \\
&= \sum_{k=1}^{K} \big(T_k \Delta_k + 3\ln\ln T_k + 8\big) \le \sum_{k=1}^{K} \big(2D(v_k) + 3\ln\ln T_k + 8\big) \le K(10 + 3\ln\ln T)
\end{aligned}$$

concluding the proof. □

Next, we extend Algorithm 2 to account for the fact that the actual feedback at time t is the random variable $r_t(X_t)$ rather than its conditional expectation $X_t D(X_t)$. The main intuition is very simple: in order to estimate $D(x)$ we divide time in blocks (called again *macrosteps*) of equal length, and build an estimate $\overline{D}(x)$ by posting the same price x within each block. In order to decide which arm i to use in each macrostep, we compute an upper confidence bound U_i on the average demand in the i-th interval, and then select the arm attaining the highest of such bounds.

Our algorithm receives as input the time horizon T, a lower bound γ on $p_{\min} = \min_i p_i$, and a confidence parameter δ. Given these parameters, the number of macrosteps is defined as the biggest $M_\gamma \in \mathbb{N}$ satisfying $T \ge M_\gamma \lceil 8\ln(\delta^{-1})/\gamma^2 \rceil$. The fraction of accepted offers of price x during the m-th macrostep (in which x is offered) is denoted by $\overline{D}_m(x)$. Our algorithm (Algorithm 3) is very similar to Algorithm 2, so we only highlight the main differences.

First, note that references to steps t are replaced by references to macrosteps m; in particular, κ_m is the number of allocated intervals after m macrosteps. In line 3, the selected arm i_m is now the one maximizing, over intervals $[a_i, b_i]$, the product $b_i U_i$. The quantity U_i is the upper confidence bound

$$U_i = \widehat{D}_m(i) + \frac{1}{b_i}\sqrt{\frac{\ln(\delta^{-1})}{N_m(i)}}$$

where $N_m(i)$ is $\lceil 8\gamma^{-2}\ln\delta^{-1}\rceil$ (if $i > 1$, which takes into account the macrostep in which interval i was allocated) plus the total number of times that i was picked in the first $m-1$ macrosteps, ignoring the steps occurring in all macrosteps when line 13 was executed. $\widehat{D}_m(i)$ is the fraction of accepted offers during these $N_m(i)$ steps. In line 7, a new valuation is detected when the difference between demands is bigger than $\gamma/2$. Finally, in line 11 a new interval is allocated if the newly discovered demand differs from all previously detected demands by at least $\gamma/2$.

Theorem 1.5. *If Algorithm 3 is run on an unknown number K of pairs $(v_1, p_1)\dots,(v_K, p_K)$ with input parameters $\gamma \le \min_k p_k$ and $\delta = T^{-2}$, then its regret satisfies*

$$R_T \le \sum_{i:\,\Delta_i>0} \frac{4\ln T}{\Delta_i} + \mathcal{O}\left(\frac{K\ln T}{\gamma^2}\ln\ln T\right) .$$

Proof. Without loss of generality, assume $M_\gamma B_\gamma = T$ where $B_\gamma \ge 8\ln(\delta^{-1})/\gamma^2$ is the length of a macrostep. Hence, for any given price $0 \le x \le 1$, Hoeffding's inequality implies $\bigl|\overline{D}(x) - D(x)\bigr| \le \gamma/4$ with probability at least $1-2\delta$. Therefore, if $D(x) - D(y) \ge \gamma$, then $\overline{D}(x) - \overline{D}(y) \ge \gamma/2$ with probability at least $1-4\delta$. Moreover, if $D(x) = D(y)$, then $\overline{D}(x) - \overline{D}(y) \le \gamma/2$ with probability at least $1-4\delta$. Since at each macrostep of the algorithm we perform at most $K+1$ comparisons between $\overline{D}(x)$ and $\overline{D}(y)$ for pairs of points x, y (lines 7 and 11), the probability that, for at least one of these comparisons, we have

$$\left(\bigl|\overline{D}_m(x) - \overline{D}_m(y)\bigr| < \frac{\gamma}{2} \;\wedge\; |D(x)-D(y)| \ge \gamma\right) \text{ or } \left(\bigl|\overline{D}_m(x) - \overline{D}_m(y)\bigr| > \frac{\gamma}{2} \;\wedge\; D(x) = D(y)\right) \tag{1.5}$$

is at most $4(K+1)\delta$. Let $\mathcal{B}$ the event that (1.5) occurs for at least one comparison in at least one macrostep. Then $\mathbb{P}(\mathcal{B}) \le 4(K+1)M_\gamma\delta$.

Assume $\mathcal{B}$ does not occur. Recall that $v_{\mu(i)}$ is the smallest valuation in $[a_i, b_i]$. Since $p_{\mu(i)} \ge \gamma$ by hypothesis, event $X_m > v_{\mu(i)}$ implies that the test in line 7 is false, and therefore line 13 is executed. Therefore, assuming event (1.5) never occurs, the macrosteps of Algorithm 3 with feedback $r_t(X_t)$ are equivalent to the steps of Algorithm 2 run with feedback $X_t D(X_t)$. In particular, Lemma 1.1 applies to the macrosteps of Algorithm 3.

Let $n_m(i)$ be the number of macrosteps (in the first $m-1$ macrosteps) where i was picked. Similarly, let $\mathrm{OS}_m(i)$ be the number of macrosteps (in the first $m-1$ macrosteps) when i was picked and $X_m > v_{\mu(i)}$. Then we have $N_m(i) = B_\gamma\bigl(n_m(i) - \mathrm{OS}_m(i)\bigr)$. Now note that $\widehat{D}_m(i)$ is the sample mean of a Bernoulli of parameter $D(v_{\mu(i)})$ because it is computed over $N_m(i)$ points sampled between a_i and $v_{\mu(i)}$. Fix a suboptimal valuation v_k and a macrostep m such that $\mu(i_m) = k$. Let $i^\star$ be such that $v^\star \in \bigl[a_{i^\star}, b_{i^\star}\bigr]$. Then,

$$\begin{aligned}
i_m \ne i^\star &\Longrightarrow b_{i^\star} U_{i^\star} \le b_{i_m} U_{i_m} \\
&\Longleftrightarrow \left(b_{i^\star}\widehat{D}_m(i^\star) + \sqrt{\frac{\ln(\delta^{-1})}{N_m(i^\star)}}\right) \le \left(b_{i_m}\widehat{D}_m(i_m) + \sqrt{\frac{\ln(\delta^{-1})}{N_m(i_m)}}\right) \\
&\Longrightarrow \left(v^\star\widehat{D}_m(i^\star) + \sqrt{\frac{\ln(\delta^{-1})}{N_m(i^\star)}}\right) \le \left(\left(v_k + \frac{2}{n_m(i_m)}\right)\widehat{D}_m(i_m) + \sqrt{\frac{\ln(\delta^{-1})}{N_m(i_m)}}\right)
\end{aligned}$$

where in the last step we used Lemma 1.5 in Section 1.8.2. Now recall that $n_m(i_m) \ge N_m(i_m)/B_\gamma$. Hence,

$$i_m \ne i^\star \Longrightarrow \left(v^\star\widehat{D}_m(i^\star) + \sqrt{\frac{\ln(\delta^{-1})}{N_m(i^\star)}}\right) \le \left(v_k\widehat{D}_m(i_m) + \frac{2B_\gamma}{N_m(i_m)} + \sqrt{\frac{\ln(\delta^{-1})}{N_m(i_m)}}\right) .$$

Observe that $\mathbb{E}\bigl[\widehat{D}_m(i^\star)\bigr] = D\bigl(v_{\mu(i^\star)}\bigr) \ge D(v^\star)$ and $\mathbb{E}\bigl[\widehat{D}_m(i_m)\bigr] = D(v_k)$. Moreover, the two quantities $\sqrt{\bigl(\ln(\delta^{-1})\bigr)/\bigl(N_m(i^\star)\bigr)}$ and $2B_\gamma/N_m(i_m) + \sqrt{\bigl(\ln(\delta^{-1})\bigr)/\bigl(N_m(i_m)\bigr)}$ play the role of upper confidence bounds for the estimates $v^\star\widehat{D}_m(i^\star)$ and $v_k\widehat{D}_m(i_m)$. Therefore, we can apply a modification of the analysis of UCB1

[4, Proof of Theorem 1] to K arms with reward expectations $v_k D(v_k)$ for $k \in \{1,\dots,K\}$, and such that the upper confidence bound for any suboptimal arm k is inflated by $2B_\gamma/N_m(i_m)$. (In fact Lemma 1.6 in Section 1.8.3 is stronger than what we need, because $v^\star$ always belongs to some interval $[a_{j^\star}, b_{j^\star}]$ but not all suboptimal valuations v_k are the smallest valuation of the interval $[a_{j_k}, b_{j_k}]$ they belong to.) In particular, recalling that $B_\gamma = 8(\ln(\delta^{-1}))/\gamma^2$ and recalling also our assumption in $\mathcal{B}$, we apply Lemma 1.6 in Section 1.8.3 with $\alpha = 16$. This gives

$$B_\gamma \, \mathbb{E}\left[\mathbb{I}\left\{\overline{\mathcal{B}}\right\} \sum_{m \colon \mu(i_m)=k} \mathbb{I}\{i_m \neq i^\star\}\right] \le 1 + \left((\delta T)^2 + \frac{64}{\gamma^2}\right) 2K\ln(\delta^{-1}) + \sum_{k \colon \Delta_k > 0} \frac{4\ln(\delta^{-1})}{\Delta_k} \, .$$

where $\Delta_k = v^\star D(i^\star) - v_k D(v_k) > 0$ and $\overline{\mathcal{B}}$ is the complement of $\mathcal{B}$.[2] Because Lemma 1.6 bounds the number of steps in which a suboptimal arm is selected, we multiplied by B_γ the right-hand side of the above, thus converting macrosteps m in steps t. The fact that we prevent the algorithm from switching arm within each macrostep is not an issue. Indeed, the proof of the Lemma works irrespective to whether the decision of pulling a different arm is made at every macrostep as opposed to every step. In particular, the proof establishes that after each suboptimal arm is selected order of $(\ln T)/\gamma^2$ times, corresponding to a constant number of macrosteps, the probability of pulling any suboptimal arm ever again becomes tiny, of order T^{-2}.

Similarly to the proof of Theorem 1.4, introduce $\mathcal{M}_k = \{m \le M_\gamma \mid v_k \in [a_{i_m}, b_{i_m}]\}$. As argued above, we may apply Lemma 1.1 to the macrosteps of Algorithm 3. Hence, bound (1.4) applies with $\mathcal{T}_k$ replaced by $\mathcal{M}_k$. Therefore, with probability at least $1 - 4(K+1)M_\gamma\delta$, the regret over the T steps (recall that we repeatedly post the same price in each step of a macrostep) is bounded by

$$\begin{aligned}
&B_\gamma \mathbb{E}\left[\sum_{m=1}^{M_\gamma} \Bigl(v^\star D(v^\star) - X_m D(X_m)\Bigr)\right] \\
&\quad\le B_\gamma \mathbb{E}\left[\sum_{k=1}^{K} \sum_{m \,:\, \mu(i_m)=k} \Bigl(v^\star D(v^\star) - v_k D(v_k)\Bigr) + \sum_{k=1}^{K} \sum_{m \in \mathcal{M}_k} \Bigl(v_k D(v_k) - X_m D(X_m)\Bigr)\right] \\
&\quad\le B_\gamma \sum_{k=1}^{K} \Delta_k \mathbb{E}\left[\mathbb{I}\left\{\overline{\mathcal{B}}\right\} \sum_{m \,:\, \mu(i_m)=k} \mathbb{I}\{i_m \neq i^\star\}\right] + T\mathbb{P}(\mathcal{B}) + B_\gamma \sum_{k=1}^{K} \bigl(3\ln\ln T_k + 8\bigr) \qquad \text{(using (1.4))} \\
&\quad\le 1 + \left((\delta T)^2 + \frac{64}{\gamma^2}\right) 2K\ln(\delta^{-1}) + \sum_{k \colon \Delta_k > 0} \frac{4\ln(\delta^{-1})}{\Delta_k} + T\mathbb{P}(\mathcal{B}) + B_\gamma K\bigl(3\ln\ln T + 8\bigr) \, . \qquad (1.6)
\end{aligned}$$

Finally, in order to bound $T\mathbb{P}(\mathcal{B}) \le 4(K+1)TM_\gamma\delta = (K+1)(T\gamma)^2\delta/(2\ln\delta^{-1})$, it is sufficient to set $\delta = T^{-2}$. □

We conclude this section by discussing the case of at most two valuations. We design an algorithm with regret of order $\log(T)/\Delta + \log(T)\log\log(T)$, which is (up to the $\log\log$ term) as if the exact values of v_1 and v_2 were known in advance! This is achieved by leveraging some properties of the smallest and the biggest valuation. For example, any offer of a price lower or equal to v_1 is deterministically accepted and all offers above v_2 are always rejected. If on the other hand a price $x \in (v_1, v_2]$ is offered, the probability that that price is accepted is exactly p_2, which is enough to reconstruct the entire distribution (p_1, p_2) on $\{v_1, v_2\}$. Furthermore, the suboptimality gap Δ is always equal to $|v_1 - p_2 v_2|$.

Other than the result itself, we believe the techniques used in designing and analyzing the algorithm could be of interest on their own. Theorem 1.9 in particular gives a way to compute a high-probability multiplicative estimate of the unknown expectation $\mu > 0$ of any $[0,1]$-valued random variable using only $\mathcal{O}\bigl(\frac{1}{\mu}\bigr)$ samples. We now state the result. All the details about the algorithm and its subroutines, their pseudocodes, and the remaining theoretical results are presented in Section 1.8.4.

[2] The factor $\mathbb{I}\{\overline{\mathcal{B}}\}$ inside the expectation is needed to reduce the problem to an instance of a standard stochastic bandit. It can be conveniently dropped in the analysis of Lemma 1.6.

Theorem 1.6. *If Algorithm 8 (see Section 1.8.4) is run with input parameter $\delta = T^{-2}$ on an unknown instance (v_1, p_1) and (v_2, p_2), then its regret satisfies $R_T = \mathcal{O}\big(\log(T)/\Delta + (\log T)(\log\log T)\big)$, where the first term is zero when $\Delta = |p_2 v_2 - v_1|$ is zero.*

1.7 Conclusions

In this work we initiated an investigation of stochastic dynamic pricing in a setting in which the distribution of buyers' private values is supported on a finite set of points in $[0, 1]$, where the number and location of these points is unknown to the seller. We studied the seller's regret in distribution-free and distribution-dependent settings, proving upper and lower bounds that show interesting connections to both the dynamic pricing setting of Kleinberg and Leighton [52] and the standard stochastic K-armed bandit setting. We also proved some preliminary results for the nonstochastic version of our model when there are two valuations but only one is unknown (Section 1.8.5).

Our work leaves some interesting questions open. Can we prove a distribution-free upper bound of order $\sqrt{KT}$ that does not depend on the locations of buyers' valuations? Can we prove a distribution-dependent upper bound without any prior knowledge at all for K larger than two? Can we obtain a $\sqrt{KT}$ regret bound in the nonstochastic setting when $K \geq 2$ and all valuations are unknown?

1.8 Deferred proofs and additional results

1.8.1 Lower Bound

In this section we prove the lower bounds (Theorems 1.1 and 1.2) stated in Section 1.4. Kleinberg and Leighton [52] showed that $R_T = \Omega(T^{2/3})$ if $T \leq K^3$ by building a distribution over a set of ε-spaced valuations $v_1, \dots, v_K \in \big[\frac{1}{2}, 1\big]$. A key technical property needed in their proof is that $\mathrm{KL}\big(\frac{1}{2v}, \frac{9}{10}\frac{1}{2v} + \frac{1}{10}\frac{1}{2(v-\varepsilon)}\big) \leq c\varepsilon^2$ for some constant c independent of ε and for all $v \geq 3/4$. We begin by showing that such construction only works if K is large compared to T.

Lemma 1.2. *For all $K \geq 1$, for all $\varepsilon \in \big(0, \frac{1}{2K}\big]$, and for all $k \in \{1, \dots, K\}$, denoting $v = \frac{1}{2} + k\varepsilon$,*

$$\mathrm{KL}\left(\frac{1}{2v} \,\Big\|\, \frac{9}{10}\frac{1}{2v} + \frac{1}{10}\frac{1}{2(v-\varepsilon)}\right) > \frac{\varepsilon}{800k} \; .$$

Proof. Fix any $K \geq 1$, $\varepsilon \in \big(0, \frac{1}{2K}\big]$, and $k \in \{1, \dots, K\}$. Denoting $v = \frac{1}{2} + k\varepsilon$,

$$\begin{aligned}
&\mathrm{KL}\left(\frac{1}{2v} \,\Big\|\, \frac{9}{10}\frac{1}{2v} + \frac{1}{10}\frac{1}{2(v-\varepsilon)}\right) \\
&\quad= \frac{1}{2v}\ln\left(\frac{\frac{1}{2v}}{\frac{9}{10}\frac{1}{2v} + \frac{1}{10}\frac{1}{2(v-\varepsilon)}}\right) + \left(1 - \frac{1}{2v}\right)\ln\left(\frac{1-\frac{1}{2v}}{1 - \left[\frac{9}{10}\frac{1}{2v} + \frac{1}{10}\frac{1}{2(v-\varepsilon)}\right]}\right) \\
&\quad= \frac{1}{2v}\ln\left(\frac{1}{1+\frac{\varepsilon}{10(v-\varepsilon)}}\right) + \frac{1}{2v}(2v-1)\ln\left(\frac{2v-1}{2v-1-\frac{\varepsilon}{10(v-\varepsilon)}}\right) \\
&\quad\geq \frac{1}{2}\left[-\ln\left(1 + \frac{\varepsilon}{5+10(k-1)\varepsilon}\right) - 2k\varepsilon\ln\left(1 - \frac{1}{10k+20k(k-1)\varepsilon}\right)\right] \qquad \left(\text{using } v = \tfrac{1}{2} + k\varepsilon \leq 1\right) \\
&\quad= \frac{1}{2}\sum_{n=1}^{+\infty}\frac{2k\varepsilon\left(\frac{1}{10k+20k(k-1)\varepsilon}\right)^n + \left(-\frac{\varepsilon}{5+10(k-1)\varepsilon}\right)^n}{n} \\
&\quad= \frac{\varepsilon}{8k\big(5+10(k-1)\varepsilon\big)^2} + \frac{\varepsilon^2}{4\big(5+10(k-1)\varepsilon\big)^2} + \frac{1}{2}\sum_{n=3}^{+\infty}\frac{2k\varepsilon\left(\frac{1}{10k+20k(k-1)\varepsilon}\right)^n + \left(-\frac{\varepsilon}{5+10(k-1)\varepsilon}\right)^n}{n}
\end{aligned}$$

$$
\begin{aligned}
&> \frac{\varepsilon}{800k} + \frac{\varepsilon^2}{400} + \varepsilon \sum_{n=3}^{+\infty} \frac{\frac{1}{k^{n-1}} \frac{1}{\left(10+20(k-1)\varepsilon\right)^n} + \frac{(-1)^n \varepsilon^{n-1}}{2} \frac{1}{\left(5+10(k-1)\varepsilon\right)^n}}{n} \\
&= \frac{\varepsilon}{800k} + \frac{\varepsilon^2}{400} + \varepsilon \sum_{n=3}^{+\infty} \frac{\frac{1}{\left(10+20(k-1)\varepsilon\right)^n} \left(\frac{1}{k^{n-1}} + (-1)^n (2\varepsilon)^{n-1}\right)}{n} \\
&> \frac{\varepsilon}{800k} + \frac{\varepsilon^2}{400} + \varepsilon \sum_{n=3}^{+\infty} \frac{\frac{1}{\left(10+20(k-1)\varepsilon\right)^n} \left(\frac{1}{k^{n-1}} - (2\varepsilon)^{n-1}\right)}{n} \\
&\geq \frac{\varepsilon}{800k} + \frac{\varepsilon^2}{400} \; . \qquad \text{(using } k \leq K \text{ and } \varepsilon \leq \tfrac{1}{2K}\text{)}
\end{aligned}
$$

This concludes the proof. □

In order to prove Theorem 1.1, we need the following lemma.

Lemma 1.3. *For all $p, q \in (0,1)$*

$$\mathrm{KL}\,(p \,\|\, q) \leq \frac{(p-q)^2}{q(1-q)} \; .$$

In particular, for all $x \in (0,1)$ and all $\alpha \in [0, 1-x)$,

$$\mathrm{KL}\big(x \,\|\, x+\alpha\big) \leq \frac{\alpha^2}{(x+\alpha)(1-x-\alpha)} \; . \tag{1.7}$$

Proof. Fix any $p, q \in (0,1)$. Using $\ln(x) \leq x - 1$ for all $x > 0$,

$$
\begin{aligned}
\mathrm{KL}(p \,\|\, q) &= p \ln\left(\frac{p}{q}\right) + (1-p)\ln\left(\frac{1-p}{1-q}\right) \leq p\frac{p-q}{q} - (1-p)\frac{p-q}{1-q} \\
&= (p-q)\left(\frac{p}{q} - \frac{1-p}{1-q}\right) = \frac{(p-q)^2}{q(1-q)} \; .
\end{aligned}
$$

□

We now restate and prove Theorem 1.1.

Theorem 1.7. *For any number of valuations $K \geq 3$ and all time horizons $T \geq K^3$ there exist K pairs $\big(v_1, p(v_1)\big), \ldots, \big(v_K, p(v_K)\big)$ such that the expected regret of any pricing strategy satisfies*

$$R_T \geq \frac{1}{375}\sqrt{KT} \; .$$

Proof. For notational convenience, fix $K \geq 2$ and define the set $\{v_0, \ldots, v_K\}$ of $K+1$ valuations by

$$v_i = \frac{1}{2} + \frac{i}{2K}, \qquad \forall i \in \{0, \ldots, K\} \; .$$

Define the distribution p_0 on $\{v_0, \ldots, v_K\}$ of the random variable V_0 by

$$\mathbb{P}(V_0 \geq v) = \sum_{i \colon v_i \geq v} p_0(v_i) = \frac{1}{2v}, \qquad \forall v \in \{v_0, \ldots, v_K\} \; .$$

With this choice of demand curve, $v\mathbb{P}(V_0 \geq v) = 1/2$, i.e., each valuation v has the same expected revenue. Furthermore, the distribution $v \mapsto p_0(v)$ satisfies the following: $p_0(v_0) = \frac{1}{K+1}$; p_0 decreases monotonically on $\{v_0, \ldots, v_{K-1}\}$, $p_0(v_{K-1}) = \frac{1}{2K-1}$, and $p_0(v_K) = 1/2$. Therefore

$$\frac{1}{2K} \leq p_0(v) \leq \frac{1}{K}, \qquad \forall v \in \{v_0, \ldots, v_{K-1}\} \; . \tag{1.8}$$

Now, for each $j \in \{\lceil K/2 \rceil, \dots, K\}$, define the distribution p_j by slightly lowering the probability of v_{j-1} and upping the probability of v_j by the same amount:

$$p_j(v_i) = \begin{cases} p_0(v_i), & i \in \{0, \dots, K\} \setminus \{j-1, j\}, \\ (1 - 4K\varepsilon) p_0(v_{j-1}), & i = j-1, \\ p_0(v_j) + 4K\varepsilon p_0(v_{j-1}), & i = j, \end{cases} \tag{1.9}$$

where $\varepsilon \in \left(0, \frac{1}{40}\right)$ is a small constant determined below. Note that if the buyers' valuations were distributed as p_j, all valuations $v \neq v_j$ would have expected revenue $\frac{1}{2}$, but v_j whould have expected revenue at least $\frac{1}{2} + \varepsilon$ because of (1.8) and (1.9). In order to define the distribution of buyers' valuations $V = (V_1, \dots, V_T)$, let J be uniformly distributed over $\{\lceil K/2 \rceil, \dots, K\}$ (that is, the set of indices $i \in \{1, \dots, K\}$ such that $v_i \geq \frac{3}{4}$). The value of J will give the "good valuation", that is the valuation with the highest expected revenue. For all t, the distribution of V_t is determined by

$$\mathbb{P}(V_t = v_i \mid J = j) = p_j(v_i), \qquad \forall i \in \{0, \dots, K\}, \forall j \in \{\lceil K/2 \rceil, \dots, K\} \ .$$

Denoting the seller's randomized strategy by $X = (X_1, \dots, X_T)$ and applying Fubini's theorem, we obtain

$$R_T = \max_{k \in \{0, \dots, K\}} \mathbb{E}_X \mathbb{E}_{J,V} \left[\sum_{t=1}^T r_t(v_k) - \sum_{t=1}^T r_t(X_t) \right] \ .$$

According to the previous identity, we can (an will!) lower bound the internal expectation assuming that the seller's strategy is deterministic. Furthermore, assume that the seller's pricing strategy only offers prices in $\{v_{\lceil K/2 \rceil}, \dots, v_K\}$ —since it is counterproductive to offer a price outside of it as all other valuations $(v_1, \dots, v_{\lceil K/2 \rceil - 1}$ in particular) have smaller expected revenues. Now let N_i be the number of times the seller offer valuation v_i,

$$N_i = \sum_{t=1}^T \mathbb{I}\{X_t = v_i\} \ .$$

By construction, each time the seller picks the "good valuation", no regret is accrued; all other times at least ε is lost. Therefore

$$\mathbb{E}_{J,V} \left[\sum_{t=1}^T r_t(v_k) - \sum_{t=1}^T r_t(X_t) \right] \geq \varepsilon \big(T - \mathbb{E}_{J,V}[N_J]\big) \ . \tag{1.10}$$

Denote by Y_t the Bernoulli random variable $\mathbb{I}\{V_t \geq X_t\}$ which is 1 if and only if the t-th buyer accepted the price offered, $Y^t = (Y_1, \dots, Y_t)$, and $Y = Y^T$. Denote by q_0 the distribution of Y if buyer's valuations were distributed as p_0 and by q_i the distribution of Y if buyer's valuations were distributed as p_i. For any deterministic function $f \colon \{0,1\}^T \to [0, M]$,

$$\begin{aligned} \mathbb{E}_V\big[f(Y) \mid J = i\big] - \mathbb{E}_0[f(Y)] &= \sum_{b^T \in \{0,1\}^T} f(b^T)\big(q_i(b^T) - q_0(b^T)\big) \\ &\leq \sum_{\substack{b^T \in \{0,1\}^T \\ q_i(b^T) > q_0(b^T)}} f(b^T)\big(q_i(b^T) - q_0(b^T)\big) \\ &\leq M \sum_{\substack{b^T \in \{0,1\}^T \\ q_i(b^T) > q_0(b^T)}} \big(q_i(b^T) - q_0(b^T)\big) \\ &\leq M \sqrt{\frac{1}{2} \mathrm{KL}(q_0 \parallel q_i)} \end{aligned}$$

where $\mathbb{E}_0$ is the expectation with respect to distribution p_0 and in the last step we used Pinsker's inequality. Let $q_i(b_t \mid b^{t-1}) = p_i\left(Y_t = b_t \mid Y_1 = b_1, \dots, Y_{t-1} = b_{t-1}\right)$ and let $q_0(b_t \mid b^{t-1})$ be defined similarly. By the chain rule of the relative entropy

$$\begin{aligned}
\mathrm{KL}(q_0 \parallel q_i) &= \sum_{t=1}^{T} q_0(b^{t-1}) \sum_{b^{t-1} \in \{0,1\}^{t-1}} \mathrm{KL}\big(q_0(b_t \mid b^{t-1}) \parallel q_i(b_t \mid b^{t-1})\big) \\
&= \sum_{t=1}^{T} q_0(b^{t-1}) \sum_{b^{t-1}:\, X_t(b^{t-1}) \neq v_i} \underbrace{\mathrm{KL}\big(q_0(b_t \mid b^{t-1}) \parallel q_i(b_t \mid b^{t-1})\big)}_{=0} \\
&\quad + \sum_{t=1}^{T} q_0(b^{t-1}) \sum_{b^{t-1}:\, X_t(b^{t-1}) = v_i} \mathrm{KL}\big(q_0(b_t \mid b^{t-1}) \parallel q_i(b_t \mid b^{t-1})\big)
\end{aligned}$$

where the relative entropy is zero when $X_t \neq v_i$ because in that case $p_i(Y_t = 1) = p_0(Y_t = 1)$. If on the other hand, $X_t = v_i$, for all $v_i \geq \frac{3}{4}$,

$$\mathrm{KL}\big(q_0(b_t \mid b^{t-1}) \parallel q_i(b_t \mid b^{t-1})\big) = \mathrm{KL}\left(\frac{1}{2v_i} \,\Big\|\, \frac{1}{2v_i} + 4K\varepsilon p_0(v_{j-1})\right) \leq 108\varepsilon^2$$

where the last inequality follows by (1.8) and $\mathrm{KL}\big(x \parallel x+\alpha\big) \leq \alpha^2 (x+\alpha)^{-1}(1-x-\alpha)^{-1}$, with $x = \frac{1}{2v_i} \in \left[\frac{1}{2}, \frac{2}{3}\right]$ and $\alpha = 4K\varepsilon p_0(v_{j-1}) \in [2\varepsilon, 4\varepsilon]$. Therefore

$$\mathrm{KL}(q_0 \parallel q_i) \leq 108\varepsilon^2 \sum_{t=1}^{T} q_0(b^{t-1}) \sum_{b^{t-1}:\, X_t(b^{t-1}) = v_i} 1 = 108\varepsilon^2 \sum_{t=1}^{T} p_0(X_t = v_i) = 108\varepsilon^2 \mathbb{E}_0[N_i] \ ,$$

where again, $\mathbb{E}_0$ is the expectation with respect to distribution p_0. This gives

$$\mathbb{E}_V[f(Y) \mid J = i] \leq \mathbb{E}_0[f(Y)] + \varepsilon M \sqrt{54 \mathbb{E}_0[N_i]} \ .$$

Then, being for any deterministic online pricing strategy the random variable N_i a deterministic function of Y, $\mathbb{E}_V[N_i \mid J = i] \leq \mathbb{E}_0[N_i] + \varepsilon T \sqrt{54\mathbb{E}_0[N_i]}$. Thus, using Jensen inequality, $\mathbb{E}_{J,V}[N_i] \leq \mathbb{E}_J \mathbb{E}_0[N_J] + \varepsilon T \sqrt{54 \mathbb{E}_J \mathbb{E}_0[N_J]}$. Using again Jensen inequality, Fubini's Theorem, and inequality (1.10),

$$\mathbb{E}_{J,V} \mathbb{E}_X \left[\sum_{t=1}^{T} r_t(v_k) - \sum_{t=1}^{T} r_t(X_t)\right] \geq \varepsilon \left(T - \mathbb{E}_J \mathbb{E}_0 \mathbb{E}_X[N_J] - \varepsilon T \sqrt{54 \mathbb{E}_J \mathbb{E}_0 \mathbb{E}_X[N_J]}\right) \ .$$

Since $\sum_{i=\lceil K/2 \rceil}^{K} N_i = T$, we also have $\sum_{i=\lceil K/2 \rceil}^{K} \mathbb{E}_0 \mathbb{E}_X[N_i] = T$. Using the fact that $K - \lceil K/2 \rceil + 1 \geq \max\{3/2, K/2\}$, this implies

$$\mathbb{E}_J \mathbb{E}_0 \mathbb{E}_X[N_J] = \frac{1}{K - \lceil K/2 \rceil + 1} \sum_{i=\lceil K/2 \rceil}^{K} \mathbb{E}_0 \mathbb{E}_X[N_i] \leq \min\left\{\frac{2}{3}, \frac{2}{K}\right\} T \ .$$

Putting everything together, we get

$$R_T \geq \varepsilon \left(T - \frac{2}{3}T - \varepsilon T \sqrt{\frac{108T}{K}}\right) = \varepsilon T \left(\frac{1}{3} - \varepsilon \sqrt{\frac{108T}{K}}\right) \ ,$$

which picking $\varepsilon = \frac{1}{6\sqrt{108}} \sqrt{K/T}$ so that $\varepsilon \sqrt{108T/K} = 1/6$, gives

$$R_T \geq \frac{1}{375} \sqrt{KT}$$

as desired. □

Algorithm 4: Cautious search

```
  Input: Time horizon $T \in \mathbb{N}$.
  Initialization: set $a \leftarrow 0$, $b \leftarrow 1$, $n \leftarrow 1$, $\varepsilon \leftarrow 1/2$.
1 for $t \in \{1, \dots T\}$ do
2     post $X_t = a + n\varepsilon$ and get feedback $Z_t = \mathbb{I}\{X_t \le v\}$;
3     if $Z_t = 1$ then                                          // undershooting
4         if $X_t + \varepsilon < b$ then update $n \leftarrow n + 1$;
5         else update $a \leftarrow X_t$, $n \leftarrow 1$, $\varepsilon \leftarrow \varepsilon^2$;      // shrink the interval
6     else if $Z_t = 0$ then                                      // overshooting
7         update $a \leftarrow X_t - \varepsilon$, $b \leftarrow X_t$, $n \leftarrow 1$, $\varepsilon \leftarrow \varepsilon^2$;   // shrink the interval
```

In summation, Even if the technique used by Kleinberg and Leighton [52] fails in our setting, it is still possible to prove an analogous lower bound by changing some key aspects of their analysis, which in turn is based on the lower bound analysis of [5]. First, valuations need to be distanced as much as possible —this is the exact opposite of their construction, where valuations were placed ε-close to each others. Second, the base distribution is only perturbed by an appropriate small constant. Third, the "good valuation" is drawn from a sensible proper subset of valuations.

1.8.2 Cautious search

Kleinberg and Leighton [52] were first to introduce a "cautious search" as an optimal algorithm for posted price with a single unknown evaluation. Similarly, our cautious search (Algorithm 4) proceeds in phases $s \in \{1, 2, \dots\}$ in which an interval $[a_s, b_s]$ (initialized to $[0, 1]$) and a step size ε_s (initialized to $1/2$) are maintained. In a given phase s of the algorithm, prices $a_s + \varepsilon_s$, $a_s + 2\varepsilon_s$, $a_s + 3\varepsilon_s$, $\dots$ are posted until one of them, say X_s, becomes bigger than the hidden evaluation (overshooting). At this point a new phase begins: the interval becomes $[a_{s+1}, b_{s+1}] = [X_s - \varepsilon_s, X_s]$, and the new step size becomes $\varepsilon_{s+1} = \varepsilon_s^2$. This process continues until the length of the interval is less than $1/T$. Then the left endpoint of the interval is picked for all remaining rounds. We now state two lemmas about the behavior of cautious search. The first one is proven in [52, Theorem 2.1].

Lemma 1.4. *The regret of Algorithm 4 satisfies* $\mathbb{E}\left[\sum_{t=1}^T r_t(v) - \sum_{t=1}^T r_t(X_t)\right] \le 3\ln\ln(T) + 8$. *Moreover, the number of overshootings is upper bounded by* $\log\log T$.

The second lemma bounds the size of the interval as a function of the number of steps.

Lemma 1.5. *For all m, the size of an interval $[a_s, b_s]$ after m steps of Algorithm 4 satisfies*

$$b_s - a_s \le \frac{2}{m} \ .$$

Proof. The worst case happens when the sequence $(b_1 - a_1, b_2 - a_2, \dots)$ of interval endpoints takes values

$$\left(1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \dots, \frac{1}{2^{2^n}}, \dots, \frac{1}{2^{2^n}}, \dots\right) \tag{1.11}$$

where the general term $1/2^{2^n}$ is repeated 2^{2^n} times. It is then sufficient to show that the inequality holds for all values before a switch. Formally, that for all $n \in \{0, 1, 2, \dots\}$

$$\frac{1}{2^{2^n}} \le \frac{2}{2 + \sum_{j=0}^n 2^{2^j}} \qquad \text{or, equivalently,} \qquad 2 + \sum_{j=0}^n 2^{2^j} \le 2 \cdot 2^{2^n} \ .$$

We prove this by induction on n. The case $n = 0$ is trivial. If the inequality holds for $n \in \{0, 1, \dots\}$, then

$$2 + \sum_{j=0}^{n+1} 2^{2^j} = 2 + \sum_{j=0}^{n} 2^{2^j} + 2^{2^{n+1}} \le 2 \cdot 2^{2^n} + 2^{2^{n+1}} = 2^{2^n} \left(2 + 2^{2^n}\right) \le 2 \cdot 2^{2^{n+1}} \, .$$

This concludes the proof. $\square$

The previous bound is unimprovable. Indeed in scenario (1.11), for all $n \in \{0, 1, \dots\}$

$$2^{2^n} < 2 + \sum_{j=0}^{n} 2^{2^j} \le 2 \cdot 2^{2^n}$$

and the second inequality is actually an equality for $n = 0$.

1.8.3 UCB with inflated confidence bounds

In this section we prove a regret bound for UCB1 run with an oracle that systematically inflates the upper confidence bounds for suboptimal arms.

Lemma 1.6. *Consider a stochastic bandit problem with K arms, i.i.d. rewards $X_t(k) \in [0,1]$ from each arm k, and average rewards $\mu_1, \dots, \mu_K$. Let $\Delta_k = \mu^\star - \mu_k$ where $\mu^\star = \mu_{i^\star}$ and $i^\star$ is the index of an optimal arm. Consider a UCB policy that at round t selects arm I_t defined by*

$$I_t = \operatorname*{arg\,max}_{k \in \{1,\dots,K\}} \left(\widehat{X}_t(k) + c\big(N_t(k), k\big)\right)$$

(ties broken arbitrarily), where $\widehat{X}_t$ is the sample average of the rewards obtained from arm k over the $N_t(k)$ times when the arm was chosen in rounds $1, \dots, t-1$ (initially, $N_1(k) = 0$ for all arms) and

$$c(s,k) = \begin{cases} \dfrac{\alpha \ln(\delta^{-1})}{\gamma^2 s} + \sqrt{\dfrac{\ln(\delta^{-1})}{s}} & \text{if } k \text{ is suboptimal,} \\ \sqrt{\dfrac{\ln(\delta^{-1})}{s}} & \text{otherwise,} \end{cases}$$

with $\alpha \ge 0$ and $c(s,k) = +\infty$ if $s = 0$. Then

$$R_T \le 1 + \left(2(\delta T)^2 + \frac{8\alpha \ln(\delta^{-1})}{\gamma^2}\right) K + \sum_{k \colon \Delta_k > 0} \frac{4 \ln(\delta^{-1})}{\Delta_k} \, .$$

Proof. Pick any suboptimal arm k and $t \ge 2$. Note that $I_t = k$ implies

$$\widehat{X}_t(i^\star) + c\big(N_t(i^\star), i^\star\big) \le \widehat{X}_t(k) + c\big(N_t(k), k\big)$$

which in turn imply

$$\left(\widehat{X}_t(i^\star) \le \mu^* - c\big(N_t(i^\star), i^\star\big)\right) \vee \left(\widehat{X}_t(k) \ge \mu_k + c\big(N_t(k), k\big)\right) \vee \left(c\big(N_t(k), k\big) > \Delta_k/2\right) .$$

Using standard Chernoff bounds, we can write

$$\begin{aligned} \sum_{t=2}^{T} \mathbb{P}\Big(\widehat{X}_t(i^\star) \le \mu^* - c\big(N_t(i^\star), i^\star\big)\Big) &\le \sum_{t=2}^{T} \mathbb{P}\Big(\exists s \in \{1, \dots, t-1\},\ \widehat{X}_t(i^\star) \le \mu^* - c(s, i^\star)\Big) \\ &\le \sum_{t=2}^{T} \sum_{s=1}^{t-1} \exp\left(-2s \frac{\ln(\delta^{-1})}{s}\right) \le T^2 \delta^2 \end{aligned}$$

and

$$\sum_{t=2}^{T} \mathbb{P}\Big(\widehat{X}_t(k) \ge \mu_k + c\big(N_t(k),k\big)\Big) \le \sum_{t=2}^{T} \mathbb{P}\Big(\exists s \in \{1,\dots,t-1\},\ \widehat{X}_t(k) \ge \mu_k + c(s,k)\Big)$$
$$\le \sum_{t=2}^{T} \mathbb{P}\left(\exists s \in \{1,\dots,t-1\},\ \widehat{X}_t(k) \ge \mu_k + \sqrt{\frac{2\ln(\delta^{-1})}{s}}\right)$$
$$\le \sum_{t=2}^{T}\sum_{s=1}^{t-1} \exp\left(-2s\frac{\ln(\delta^{-1})}{s}\right) \le T^2\delta^2 \ .$$

It remains to control $\mathbb{I}\left\{c\big(N_t(k),k\big) > \Delta_k/2\right\}$ when $I_t = k$. We now show that

$$\sum_{t=2}^{T} \mathbb{I}\left\{c\big(N_t(k),k\big) > \Delta_k/2\right\} \le 4\left(\frac{2\alpha}{\gamma^2\Delta_k} + \frac{1}{\Delta_k^2}\right)\ln(\delta^{-1}) \ .$$

If k is chosen $s > 0$ times in the first $t-1$ steps, then $N_t(k) = s$. Thus $c(s,k) > \Delta_k/2$ implies

$$\frac{\alpha\ln(\delta^{-1})}{\gamma^2 s} + \sqrt{\frac{\ln(\delta^{-1})}{s}} > \frac{\Delta_k}{2} \ . \tag{1.12}$$

We now prove that s must be smaller than

$$4\left(\frac{2\alpha}{\gamma^2\Delta_k} + \frac{1}{\Delta_k^2}\right)\ln(\delta^{-1})$$

for this to happen. If $\alpha = 0$ this is trivially true. To see that this still true for $\alpha > 0$, note that with this assumption (1.12) is equivalent to

$$\sqrt{\frac{\ln(\delta^{-1})}{s}} > \frac{-1 + \sqrt{1 + 2\Delta_k\alpha/\gamma^2}}{2\alpha/\gamma^2} \ .$$

Set $x = 2\Delta_k\alpha/\gamma^2 > 0$ so that the above can be rewritten as

$$\sqrt{\frac{\ln(\delta^{-1})}{s}} > \frac{\Delta_k\left(\sqrt{1+x}-1\right)}{x}$$

or, squaring both sides,

$$\frac{s}{\ln\delta^{-1}} < \frac{x^2}{\Delta_k^2\left(\sqrt{1+x}-1\right)^2} \ .$$

We now prove that

$$\frac{x^2}{\Delta_k^2\left(\sqrt{1+x}-1\right)^2} \le \frac{4}{\Delta_k^2}(x+1) \ .$$

Indeed, the above is equivalent to

$$\frac{x}{\sqrt{1+x}-1} \le 2\sqrt{1+x}$$

which holds because

$$\frac{x}{\sqrt{1+x}-1} = \frac{\left(\sqrt{1+x}+1\right)\left(\sqrt{1+x}-1\right)}{\sqrt{1+x}-1} \le 2\sqrt{1+x} \ .$$

Setting $\delta = T$, the regret is therefore bounded as follows

$$R_T \le 1 + \sum_{k\,:\,\Delta_k>0} \Delta_k \sum_{t=2}^{T} \mathbb{P}(I_t = k) \le 1 + 2KT^2\delta^2 + \frac{8\alpha K}{\gamma^2}\ln(\delta^{-1}) + \sum_{k\,:\,\Delta_k>0} \frac{4\ln(\delta^{-1})}{\Delta_k} \ .$$

This concludes the proof. □

Algorithm 5: Noisy Cautious Search

```
   Input: confidence parameter $\delta \in (0,1)$, valuation index $i \in \{1,2\}$, lower bound $\gamma_i \in (0,1)$.
   Initialization: set $a \leftarrow 0$, $b \leftarrow 1$.
1  for $s \in \{0,1,\dots,\lceil \log_2 \log_2 T \rceil\}$ do                                  // phases
2      set $n \leftarrow 1$, $\varepsilon_s \leftarrow 2^{-2^s}$, $\overline{D} \leftarrow 1$;
3      while $(a + n\varepsilon_s < b) \wedge \big[(i = 1 \wedge \overline{D} = 1) \vee (i = 2 \wedge \overline{D} > 0)\big]$ do
4          offer price $a + n\varepsilon_s$ for $\lceil \ln(\delta)/\ln(1-\gamma_i) \rceil$ rounds;      // a macrostep
5          update $n \leftarrow n+1$ and the sample mean $\overline{D}$ of $D\big(a + (n-1)\varepsilon_s\big)$;
6      update $a \leftarrow a + (n-1)\varepsilon_s$, $b \leftarrow a + n\varepsilon_s$;
7  offer $a$ for all remaining rounds;
```

1.8.4 Two valuations

In this section we present all key results related to subroutines of Algorithm 8 and give a formal proof of Theorem 1.6.

Noisy Cautious Search

This procedure is a variant of the cautious search described in Section 1.8.2. It identifies the location of a valuation v_i with high probability and low regret whenever a lower bound γ_i on its probability p_i is known in advance. During the search, each price is posted for $\lceil \ln(\delta)/\ln(1-\gamma_i) \rceil$ times in a row, where δ is a confidence parameter. We call such a sequence of consecutive rounds a *macrostep*. For $i = 1$, we say that a macrostep is a *failure* if at least one price is rejected, it is a *success* if all prices are accepted, and the algorithm makes a *mistake* if the macrostep is a success but the price offered is strictly bigger than v_1. For $i = 2$, we say that a macrostep is a *failure* if no price is accepted, it is a *success* if at least one price is accepted, and the algorithm makes a *mistake* if the macrostep is a failure but the price offered is at most v_2.

The Noisy Cautious Search for a valuation v_i proceeds in phases and begins by offering $1/2$ during the first macrostep. During each phase $n \geq 0$, if the last macrostep was a success, the price offered is increased by 2^{-2^n}. As soon as a macrostep is a failure, phase n ends and phase $n+1$ begins by offering the price of the last successful macrostep, plus $2^{-2^{n+1}}$. After $\lceil \log_2 \log_2 T \rceil$ phases, the price of the last successful macrostep is offered for all remaining rounds.

Lemma 1.7. *The Noisy Cautious Search for v_i with parameters i, δ, γ_i satisfies the following:*

1. *the price offered during each macrostep m is $2/m$-close to v_i with probability at least $1 - m\delta$;*
2. *the total reward accumulated by the end of macrostep m is at least*

$$\big(m v_i D(v_i) - 3(\ln \ln T) - 8\big) \frac{\ln \delta}{\ln(1-\gamma_i)}$$

   with probability at least $1 - m\delta$.

Proof. Claim 1 follows by Lemma 1.5 and the fact that the probability of making a mistake during each macrostep is at most δ by Chernoff inequality for Bernoulli random variables. Similarly, claim 2 follows by Lemma 1.4 and, again, Chernoff inequality. □

Capped Mean Estimation

We begin this section by providing a method to find a high-confidence multiplicative estimate of the expectation μ of any $[0,1]$-valued random variable, using only $\mathcal{O}\big(\ln(1/\delta)/\mu\big)$ samples. Most notably, the expectation μ need *not* be known in advance. With our novel technique, we improve upon Berthet and Perchet [9, Lemma 13], that proved a similar risult using $\mathcal{O}\big(\ln(1/\delta)/\mu^2\big)$ samples. This result will be pivotal for our analysis and we believe it will also be valuable in its own right. For any set $X_1, \dots, X_T$ of random variables, we denote by

$$\overline{X}_t = \frac{1}{t} \sum_{s=1}^{t} X_s \qquad \text{and} \qquad S_t^2 = \frac{1}{t-1} \sum_{s=1}^{t} \left(X_s - \overline{X}_t \right)^2$$

the *sample mean* and the *sample variance* of the first t random variables. The following result is a straightforward consequence of the empirical Bernstein bound and the confidence bound for standard deviation proven in [59, Theorems 4, 10].

Theorem 1.8. *Let $X_1, \dots, X_T$ be a set of $[0,1]$-valued i.i.d. random variables with expectation μ and standard deviation σ. For all $\delta \in (0,1)$ and all $t \in \{2, \dots, T\}$, the two following conditions hold simultaneously with probability at least $1 - 3\delta$*

$$\left| \overline{X}_t - \mu \right| \le \sqrt{2} S_t \left(\frac{\ln(1/\delta)}{t} \right)^{1/2} + \frac{7}{3} \frac{\ln(1/\delta)}{t-1} \qquad \textit{and} \qquad S_t \le \sigma + \sqrt{2} \left(\frac{\ln(1/\delta)}{t-1} \right)^{1/2} .$$

We can now prove our multiplicative mean estimation theorem.

Theorem 1.9 (Multiplicative mean estimation)**.** *Let $X_1, \dots, X_T$ be a set of $[0,1]$-valued i.i.d. random variables with expectation $\mu > 0$ and standard deviation σ. For all $\delta \in (0,1)$ and all $\alpha \ge 0$, if $T \ge t_0$, where*

$$t_0 = \left\lceil \frac{\alpha+2}{3\mu} \ln\left(\frac{1}{\delta}\right) \left(\sqrt{9\alpha^2 + 114\alpha + 192} + 3\alpha + 19 \right) \right\rceil + 2 = \mathcal{O}\left(\frac{\alpha^2}{\mu} \ln \frac{1}{\delta} \right)$$

and $\tau = \tau(T, \delta, \alpha)$ is the smallest time $t \in \{2, \dots, T\}$ such that

$$\frac{\overline{X}_t}{\alpha+1} \ge \sqrt{2} S_t \left(\frac{\ln(1/\delta)}{t} \right)^{1/2} + \frac{7}{3} \frac{\ln(1/\delta)}{t-1} \tag{1.13}$$

then, with probability at least $1 - 3(T-1)\delta$,

1. *$\tau \le t_0$,*
2. *for all $t \in \{2, \dots, T\}$ such that (1.13) holds,*

$$\left(\frac{\alpha}{\alpha+1} \right) \overline{X}_t < \mu < \left(\frac{\alpha+2}{\alpha+1} \right) \overline{X}_t . \tag{1.14}$$

Proof. Denote for all $t \in \{2, \dots, T\}$, $c_t = \sqrt{2\, S_t^2 \ln(1/\delta)/t} + (7/3)\ln(1/\delta)/(t-1)$. By Theorem 1.8, the *good* event

$$G = \left\{ \forall t \in \{2, \dots, T\}, \quad \overline{X}_t - c_t < \mu < \overline{X}_t + c_t \quad \text{and} \quad S_t \le \sigma + \sqrt{2 \ln(1/\delta)/(t-1)} \right\}$$

has probability $\mathbb{P}(G) \ge 1 - 3(T-1)\delta$. For all outcomes in G and all $t \in \{2, \dots, T\}$,

$$\overline{X}_t < (\alpha+1) c_t \iff \mu - c_t < \overline{X}_t < (\alpha+1) c_t \implies \mu < (\alpha+2) c_t \implies t < t_0$$

hence $\tau \le t_0$. This implies that for all outcomes in G and all $t \in \{1, \dots, T\}$ such that $\overline{X}_t \ge (\alpha+1) c_t$,

$$\left(\frac{\alpha}{\alpha+1} \right) \overline{X}_t = \overline{X}_t - \frac{\overline{X}_t}{\alpha+1} \le \overline{X}_t - c_t < \mu < \overline{X}_t + c_t \le \overline{X}_t + \frac{\overline{X}_t}{\alpha+1} = \left(\frac{\alpha+2}{\alpha+1} \right) \overline{X}_t.$$

□

The following capped version of the previous theorem interrupts the process if during the multiplicative mean estimation it is learned that μ is smaller than some threshold parameter θ.

Corollary 1.1 (Capped Mean Estimation)**.** *For any threshold parameter $\theta \in [0,1]$, under the same assumptions of Theorem 1.9, define $\tau_\theta = \min\{\tau, t_\theta\}$, where*

$$t_\theta = \left\lceil \frac{\alpha+2}{3\theta} \ln\left(\frac{1}{\delta}\right) \left(\sqrt{9\alpha^2 + 114\alpha + 192} + 3\alpha + 19 \right) \right\rceil + 2 = \mathcal{O}\left(\frac{\alpha^2}{\theta} \ln \frac{1}{\delta} \right) .$$

With probability at least $1 - 3(T-1)\delta$,

```
Algorithm 6: Capped Mean Estimation
  Input: $x_1, x_2, \ldots \in [0,1]$, $\theta \in [0,1]$, $\delta \in (0,1)$, $\rho \in \{0,1\}$.
  Initialization: set $t \leftarrow 3$ and $\widehat{D}_s = (1-\rho)\mathbb{I}\{V_s \geq x_s\} + \rho(1 - \mathbb{I}\{V_s \geq x_s\})$ for all $s$.
1 offer $x_1$ and $x_2$ once each;
2 set $\overline{D} \leftarrow \frac{1}{2}\sum_{s=1}^{2} \widehat{D}_s$ and $S^2 \leftarrow \sum_{s=1}^{2} \big(\widehat{D}_s - \overline{D}\big)^2$;
3 while $\big[t \leq \lceil 40\ln(1/\delta)/\theta \rceil + 2\big] \wedge \big[\overline{D} < \sqrt{8S^2\ln(1/\delta)/t} + (14/3)\ln(1/\delta)/(t-1)\big]$ do
4     offer price $x_t$ once;
5     update $\overline{D} \leftarrow \big(\overline{D}(t-1) + \widehat{D}_t\big)/t$, $S^2 \leftarrow \big(S^2(t-2) + (\widehat{D}_t - \overline{D})^2\big)/(t-1)$, and $t \leftarrow t+1$;
6 if $t > \lceil 40\ln(1/\delta)/\theta \rceil + 2$ then return that $\mu \leq \theta$;
7 else return $\overline{D}/2$;
```

1. *if $\tau_\theta = \tau$, then for all $t \in \{2, \ldots, T\}$ such that (1.13) holds, inequalities (1.14) also hold;*
2. *if $\tau_\theta = t_\theta$, then $\mu \leq \theta$.*

Our Capped Mean Estimation is defined as the Capped Mean Estimation of the demand curve (or one minus the demand curve if $\rho = 1$) at a given sequence of prices[3] $x_1, x_2, \ldots$, with threshold $\theta \in [0,1]$ (where $1/\theta$ is interpreted as ∞ when $\theta = 0$), confidence parameter $\delta \in (0,1)$, reverse parameter ρ (that regulates if $D(x_1)$ or $1 - D(x_1)$ is being estimated) and $\alpha = 1$ (Algorithm 6).

Variant: Joint Capped Mean Estimation

We call (w, θ, δ)-Joint Capped Mean Estimation a variant of Algorithm 6 in which $x_t = w$ for all t and estimations for both $\rho = 0$ and $\rho = 1$ are carried on at the same time; i.e., where both $\overline{D}$ (sample mean for $\rho = 0$) and $\overline{D}' = 1 - \overline{D}$ (sample mean for $\rho = 1$), as well as their respective sample variances S^2 and $(S')^2$ are maintained; the condition $\big[\overline{D} \leq \sqrt{8S^2\ln(1/\delta)/t} + (14/3)\ln(1/\delta)/(t-1)\big]$ in the **while** loop is replaced by

$$\big(A \vee A'\big) = \left(\left[\overline{D} < \sqrt{\frac{8S^2}{t}\ln\frac{1}{\delta}} + \frac{14}{3(t-1)}\ln\frac{1}{\delta}\right] \vee \left[\overline{D}' < \sqrt{\frac{8(S')^2}{t}\ln\frac{1}{\delta}} + \frac{14}{3(t-1)}\ln\frac{1}{\delta}\right] \right)$$

and at the end, we return $\overline{D}/2$ (resp., $\overline{D}'/2$) and we say that $D(w)$ (resp., $1 - D(w)$) is *well-estimated* if and only if A (resp., A') is false; if A (resp., A') is true we return that $D(w)$ (resp., $1 - D(w)$) is at most θ.

Variant: Capped Mean Estimation on Noisy Cautious Search

With a slight abuse of notation, we say that a (θ, δ, ρ)-Capped Mean Estimation is run on a (δ, i, γ_i)-Noisy Cautious Search if $x_1, x_2, \ldots$ are the prices offered during the first successful macrosteps of a (δ, i, γ_i)-Noisy Cautious Search run for $\Theta\big(\frac{1}{D(x_1)}\big)$ macrosteps (resp., $\Theta\big(\frac{1}{1-D(x_1)}\big)$ macrosteps); i.e., while the Noisy Cautious Search proceeds, an increasingly accurate estimate $\widehat{p}$ of $D(x_1)$ (resp., $1 - D(x_1)$) is maintained at the same time using samples from successful macrosteps; as soon as the stopping criterion for the Capped Mean Estimation is met, the estimation stops while the Noisy Cautious Search proceeds until it reaches $\lceil 6/\widehat{p} \rceil$ macrosteps, at which point the whole process ends returning $\widehat{p}$ and the price $\widehat{v}_i$ offered during the last succesful Noisy Cautious Search macrostep.

Cautious Mean Estimation

The main idea of this section is that the problem for $K = 2$ is completely solved by determining v_1, v_2, and p_2. This suggests that computing an high-confidence estimate p_2 once a value $w \in (v_1, v_2]$ is located might be a good idea. Sadly, it is not. The problem with this approach is that if p_2 is very small an arbitrary high

[3]This algorithm is only used for prices $x_1, x_2, \ldots$ such such that $D(x_s) = D(x_t)$ for all s, t.

Algorithm 7: Cautious Mean Estimation

```
Input: price $w \in [0,1]$, confidence parameter $\delta \in (0,1)$.
1  run a $(w, 2^{-2}, \delta)$-Joint Capped Mean Estimation, returning $\widehat{p}_1, \widehat{p}_2$;
2  if $D(w)$ and $1 - D(w)$ are both well-estimated then              // $1/4 \le p_1, p_2 \le 3/4$
3      return $\widehat{p}_1, \widehat{p}_2$;
4  else if $1 - D(w)$ is well-estimated then                         // $p_1 > 3/4$
5      for $s \in \{2, 3, \dots\}$ do
6          offer $2^{-s}$ for $\lceil \ln(\delta)/\ln(3/4) \rceil$ rounds;
7          if all offers are accepted then break and return that $v_1$ is optimal;
8          else
9              continue the Joint Capped Mean Estimation with new parameters $w, 2^{-(s+1)}, \delta$;
10             if $p_1$ and $p_2$ are both well-estimated then break and return $\widehat{p}_1, \widehat{p}_2$;
11 else if $D(w)$ is well-estimated then                             // $p_2 > 3/4$
12     run $(0, \delta, 1)$-Capped Mean Estimation on $\big(\delta, 2, \frac{3}{4}\big)$-Noisy Cautious Search, returning $\widehat{p}_1, \widehat{v}_2$;
13     offer $\widehat{v}_2 \widehat{q}_2 - \widehat{p}_1$ for $\lceil \ln(1/\delta)/\widehat{p}_1 \rceil$ rounds, where $\widehat{q}_2 \leftarrow 1 - \widehat{p}_1$;
14     if at least one offer is rejected then return that $v_2$ is optimal;
15     else return $\widehat{p}_1, \widehat{p}_2$;
```

regret may be incurred in doing so. On the other hand, the more evidence is gathered that p_2 is very small, the less likely it is that v_2 is optimal. For these and other more subtle reasons, a great deal of caution is needed in order to obtain estimate of p_2 that is just good enough to use.

The algorithm we present for dealing with these issues is called Cautious Mean Estimation and it receives as an input a price $w \in (v_1, v_2]$ (i.e., that can be used to estimate p_2), as well as a confidence parameter δ. The routine begins by determining if p_1 and p_2 are both bigger than $1/4$ by using a Joint Capped Mean Estimation and invoking Corollary 1.1. If this is true, it simply returns the estimates of p_1 and p_2 to the main routine; otherwise it behaves differently depending on which one is true: $p_2 \le 1/4$ or $p_2 \ge 3/4$, which can be checked invoking again Corollary 1.1. If $p_2 \le 1/4$, it proceeds in phases. In each phase s, it checks if $v_1 \ge 2^{-s}$ by offering 2^{-s} a small number of times, in which case it halts returning that v_1 is the optimum. If it is not, it determines if p_1 and p_2 are bigger than $2^{-(k+1)}$ by using one more time Corollary 1.1, in which case it returns their estimates to the main routine. If they are not, it moves on to phase $k+1$. If on the other hand p_2 was bigger than $3/4$, it performs a Noisy Cautious Search for v_2, while at the same time collecting samples to estimate p_1, returning estimates $\widehat{v}_2$ and $\widehat{p}_1$. Then it first checks if $v_1 \le \widehat{v}_2(1-\widehat{p}_1) - \widehat{p}_1$ by posting the latter for $\lceil \ln(1/\delta)/\widehat{p}_1 \rceil$ rounds. If the test is positive, it halts returning that v_2 is the optimum. Otherwise it returns $\widehat{p}_1$ and $\widehat{p}_2$ to the main routine.

Lemma 1.8. *For all $w \in (v_1, v_2]$ and all $\delta \in (0,1)$, the Cautious Mean Estimation run with parameters w, δ satisfies the following with probability at least $1 - (15T - 13)\delta$:*

1. *if the algorithm returns that v_1 or v_2 is optimal, then it is correct;*
2. *if the algorithm returns $\widehat{p}_1$ and $\widehat{p}_2$, then both satisfy $p_i/3 < \widehat{p}_i < p_i$;*
3. *the regret of the algorithm it at most $(13)^2 \ln(1/\delta) + 6$.*

Proof. By definition of Joint Capped Mean Estimation, line 7 lasts for at most $\lceil 160 \ln(1/\delta) \rceil + 2$ rounds, which upper bounds the regret accrued during those time steps. Denote G the *good* event in which which items 1 and 2 of Corollary 1.1 hold simultaneously for both the estimate of p_1 and p_2. To prove the result, we can (and do!) restrict our analysis to *good* outcomes, i.e., outcomes belonging in G. Indeed, Corollary 1.1 implies that one and only one of the three conditions at lines 2, 4, and 11 is executed with probability at least $\mathbb{P}(G) \ge 1 - 6(T-1)\delta$ and we will show that the result holds in all three cases.

If the condition at line 2 is true, then the result follows immediately by Corollary 1.1.

Assume now that the condition at line 4 is true and fix $k \in \mathbb{N}$ such that $2^{-k} \le \max\{v_1, p_2\} \le 2^{-(k-1)}$. Note that if $v_1 \ge p_2$, the loop at line 5 will break with probability at least $1-\delta$ (by Chernoff inequality) at line 7 as soon as $s = k$; this proves point 1 for v_1. If on the other hand $v_1 < p_2$, the loop will break with probability at least $1-6(T-1)\delta$ (by Corollary 1.1) at line 10 as soon as $s = k-1$; this proves point 2. In any case, then, at most $k-1$ cycles of the loop are performed with probability at least $1-(6T-5)\delta$. If $s \le k$, line 6 is performed at most $k-1$ times and since the cost of sampling is at most v_1 (if v_1 is optimal) or p_2 (if v_2 is optimal), than the total regret accrued by executing line 6 is at most $(k-1)\lceil \ln(\delta)/\ln(3/4)\rceil \max\{v_1, p_2\} \le (e\ln 2)^{-1}\lceil \ln(\delta)/\ln(3/4)\rceil$, where we used $x\log_2(1/x) \le (e\ln 2)^{-1}$, for all $x > 0$. On the other hand, by the end of phase k the Joint Capped Mean Estimation at lines 7, 9 has offered w for at most $\lceil 2^{k+1}40\ln(1/\delta)\rceil + 2$ accruing at most $40\ln(1/\delta) + 3$ regret. This proves point 3.

Finally, consider the case in which the condition at line 11 is true. The Noisy Cautious Search at line 12 stops after at most $\lceil 40\ln(1/\delta)/p_1\rceil + 2$ rounds, returning $\widehat{p}_i \in (p_i/3, p_i)$, with probability at least $1-(7T-6)\delta$ by the fact that it makes a mistake with probability at most δ and Theorem 1.9. This proves point 2. If v_2 is optimal, Lemma 1.7 shows that the regret of the Noisy Cautious Search is at most $\big(3(\ln\ln T) + 8\ln(1/\delta)\big)\ln(4/3)$ with probability at least $1-T\delta$. If v_1 is optimal, the additional regret is at most $\big(\lceil 40\ln(1/\delta)/p_1\rceil + 2\big)(v_1 - wp_2) \le 40\ln(1/\delta) + 3$.

Consider now lines 13-14. Since $p_1/3 < \widehat{p}_1 < p_1$, then $p_2 < \widehat{q}_2 < p_2 + (2/3)p_1$. Furthermore, $v_2 - p_1 \le \widehat{v}_2 \le v_2$ with probability at least $1-T\delta$ by Lemma 1.7. If the test at line 14 is true, then $v_1 < v_2p_2$ and v_2 is optimal with probability at least $1-\delta$; this proves point 1 for v_2. To compute the regret accumulated at line 13, assume first that v_1 is optimal; then necessarily $v_1 \ge \widehat{v}_2\widehat{q}_2 - \widehat{p}_1$ and the regret of line 13 is at most $(v_1 - \widehat{v}_2\widehat{q}_2 + \widehat{p}_1)\lceil 3\ln(1/\delta)/p_1\rceil \le 9\ln(1/\delta) + 3$. If on the other hand v_2 is optimal, then the regret of line 13 is at most $(p_2v_2 - \widehat{v}_2\widehat{q}_2 + \widehat{p}_1)\lceil 3\ln(1/\delta)/p_1\rceil \le 6\ln(1/\delta) + 2$. This proves point 3 and concludes the proof. □

2-UCB

This subroutine is a slightly modified version of Algorithm 3. The only differences are that two feasible intervals are initialized at the beginning, each valuation v_i gets a personalized number of rounds $\lceil 8\ln(\delta)/\ln(1-\gamma_i)\rceil$ at line 6, and the test at line 11 need not be executed as it is known in advance that $K = 2$.

The following result is a straightforward adaptation of Theorem 1.5. As such, the proof is omitted.

Lemma 1.9. *If $\Delta = |p_2v_2 - v_1|$, $\gamma_1 \le p_1$, $\gamma_2 \le p_2$, and 2-UCB run with $\delta = T^{-2}$, it incurs a regret*

$$\mathcal{O}\left(\frac{\ln T}{\Delta} + (\ln T)(\ln\ln T)\left(\frac{\alpha_1}{-\ln(1-\gamma_1)} + \frac{\alpha_2}{-\ln(1-\gamma_2)}\right)\right) ,$$

where $(\alpha_1, \alpha_2) = (v_1 - v_1p_2, v_1)$ if v_1 is optimal, $(\alpha_1, \alpha_2) = (v_2p_2 - v_1p_2, p_2v_2)$ if v_2 is optimal, and the first term is absent if $\Delta = 0$.

Proof of Theorem 1.6

We finally have all the instruments to prove Theorem 1.6, that we restate for completeness.

Theorem 1.6. *If Algorithm 8 is run on two unknown pairs (v_1, p_1) and (v_2, p_2) with input parameter $\delta = T^{-2}$, then its regret satisfies*

$$R_T = \mathcal{O}\left(\frac{\log T}{\Delta} + (\log T)(\log\log T)\right) ,$$

where the first term is absent if $\Delta = |p_2v_2 - v_1|$ is zero.

Proof. Putting together the proofs of all previous lemmas, the probability of making a mistake in at least a test of at least a routine is upper bounded by $\mathcal{O}(T\delta)$. For this reason, we can (an do) assume that no mistakes happen. We divide the proof into three different cases.

Algorithm 8:

```
Input: Confidence parameter $\delta \in (0,1)$.
1  run a Binary Search, returning $[a_1, a_2]$;                                  // phase 1
2  run a Capped Mean Estimation of the demand at $a_2$ with parameter $\theta = a_1$, returning $\widetilde{p}_2$;
3  if $\widetilde{p}_2 > 0$ then set $w \leftarrow a_2$ ;
4  else
5      offer price $a_1$ until it is rejected;                      // check if $a_1 < v_1 \le v_2 < a_2$
6      set $w \leftarrow a_1$;
7  run a Cautious Mean Estimation of the demand at $w$, returning $\widehat{p}_1$ and $\widehat{p}_2$;    // phase 2
8  if the Cautious Mean Estimation was halted because $v_1$ or $v_2$ is the obvious optimum then
9      run a Cautious Search for the optimal valuation with lower bound 1/2;
10 else run 2-UCB with parameters $\gamma_1 = \widehat{p}_1$ and $\gamma_2 = \widehat{p}_2$ ;      // shrink the interval
```

Case 1 Assume that during phase 1 all offers of a_2 are rejected and all offers of a_1 are accepted. Consider the following four subcases. If $a_1 \le v_1 \le v_2 \le a_2$, the regret is at most $\mathcal{O}(\log T)$. Assume now that $v_1 \le a_1 \le v_2 \le a_2$. If v_2 is optimal, then the regret is at most $\mathcal{O}\big(\log(T)/a_1\big) = \mathcal{O}\big(\log(T)/\Delta\big)$. If v_1 is optimal, then the regret is at most $\mathcal{O}\big((v_1 - a_1 p_2)T + v_1 \ln(T)/a_1\big) = \mathcal{O}\big(a_1 p_1 T + \ln(T)\big)$. Note that this case only happens with probability $p_2^{\mathcal{O}(T - \ln(T)/a_1)}$, which is at least $1/T$ only if $p_1 = \mathcal{O}\big(\frac{\log T}{T - \log(T)/a_1}\big)$. Now, if $a_1 = \Omega\big(\log(T)/T\big)$ then the regret is at most $\mathcal{O}(\log T)$; otherwise it is at most $\mathcal{O}(\log T)$ because v_1 is small. If $a_1 \le v_1 \le a_2 \le v_2$, the regret is at most $\mathcal{O}\big((\max\{v_1, p_2 v_2\} - a_1)T + \max\{v_1, p_2 v_2\}\log(T)/a_1\big)$. Since all offers of a_2 were rejected, Corollary 1.1 implies that $p_2 \le a_1$, then $p_2 v_2 \le a_1 \le v_1$, hence v_1 is optimal. The regret is therefore at most $\mathcal{O}(\log T)$. Finally, assume that $v_1 \le a_1 \le a_2 \le v_2$. Combining the same arguments as above, v_1 is optimal but p_1 is small and the total regret is at most $\mathcal{O}(\log T)$.

Case 2 Assume that during phase 1 some offers of a_2 are accepted. Corollary 1.1 implies that the first Capped Mean Estimation lasts at most $\mathcal{O}\big(\log(T)/\max\{a_1, p_2\}\big)$ rounds, hence its regrets is at most $\mathcal{O}(\log T)$. Lemma 1.8 implies that the Cautious Mean Estimation has a regret at most $\mathcal{O}(\log T)$. If the cautious mean estimation is halted returning that v_1 or v_2 is optimal, then Lemma 1.7 implies that the regret is at most $\mathcal{O}\big((\log T)(\log\log T)\big)$. Assume now that the cautious mean estimation returns $\widehat{p}_1, \widehat{p}_2$. By construction, if $p_2 \le 1/4$, then necessarily $v_1 \le 2p_2$, thus $\Delta \le 2p_2$. On the other end, if $p_2 \ge 3/4$, then necessarily $v_1 \ge p_2 v_2 - 2p_1$ thus $\Delta \le 2p_1$ if v_2 is optimal. Using Lemma 1.9 and plugging in the above upper bounds gives the result.

Case 3 Assume that during phase 1 all offers of a_2 are rejected and some offers of a_1 are rejected. The proof of this case is the same as the previous one, except that sampling a_2 has an extra regret cost. If v_1 is optimal, then the additional regret is at most $\mathcal{O}\big(v_1 \log(T)/a_1\big) = \mathcal{O}(\ln T)$ because $v_1 < a_1$. Finally, assume that v_2 is optimal. If $v_2 \le a_2$, the additional cost is at most $\big(\ln(T)/a_1\big)p_2 v_2 = \mathcal{O}(\ln T)$. If $a_2 < v_2$, then $p_2 \le a_1$ by Corollary 1.1 and the additional cost is at most $\mathcal{O}\big((p_2 v_2 - p_2 a_2)\log(T)/a_1\big) = \mathcal{O}(\log T)$. □

1.8.5 Nonstochastic dynamic pricing: some initial results

In this section we present some initial results for the nonstochastic setting; namely, when the sequence $V_1, V_2, \ldots$ is deterministic rather than stochastic. This setting was studied in [52] without the restriction that each V_t belongs to a common finite set of valuations. We show an upper bound of $\mathcal{O}(\sqrt{T})$ on the regret in the simple case when $V_t \in \{v_1, v_2\}$ for all t (with $0 \le v_1 \le v_2 \le 1$) and v_2 is known. Note that this is not significantly improvable, as a matching lower bound of $\Omega(\sqrt{T})$ can be proven in the stochastic setting even when v_1 and v_2 are both known. To see that, consider $v_1 = \frac{1}{2}$ and $v_2 = \frac{3}{4}$ with $D(v_2) = \frac{2}{3} \pm \varepsilon$ for $\varepsilon < \frac{1}{3}$, so that v_1 has constant revenue $\frac{1}{2}$, and v_2 has expected revenue $\frac{1}{2} \pm \frac{3}{4}\varepsilon$. We can now adapt the argument in the proof of the nonstochastic bandit lower bound of [5] for the two equiprobable scenarios $D(v_2) = \frac{2}{3} + \varepsilon$ and

$D(v_2) = \frac{2}{3} - \varepsilon$. This allows us to conclude that in the first T rounds any algorithm suffers regret of order εT unless v_2 is played at least ε^{-2} times. Choosing $\varepsilon = T^{-1/2}$ gives the desired bound.

The algorithm achieving regret $R_T = \mathcal{O}\bigl(\sqrt{T}\bigr)$ uses the nonstochastic bandit algorithm Exp3 [5] fed with losses $\ell_t(x) = 1 - r_t(x)$. Since

$$\sum_{t=1}^{T}\Bigl(r_t(x) - r_t(X_t)\Bigr) = \sum_{t=1}^{T}\Bigl(\ell_t(X_t) - \ell_t(x)\Bigr)$$

always hold, bounding the regret of Exp3 defined with respect to losses is equivalent to bounding the regret with respect to revenues. However, as only v_2 is known, we run Exp3 using two actions: v_2 and an action, called b, that starts at v_2 and converges to v_1 during the execution of Exp3. In particular, we decrease b by steps of length $T^{-1/2}$ whenever b is played and rejected. Clearly, b stops moving as soon as $b \in \bigl(v_1 - T^{-1/2}, v_1\bigr]$, which is good enough to bound Exp3's future regret. In order to bound the regret incurred while $b > v_1$ holds, note that as long as $b > v_1$ is true, we have: $\ell_t(b) < \ell_t(v_1)$ when $V_t = v_2$, and $\ell_t(b) > \ell_t(v_1)$ when $V_t = v_1$. The problem is that we can not bound deterministically the smallest time t such that $b \le v_1$, as this depends on the buyers' choices and the algorithm's random sequence of actions. On the other hand, we know that $\ell_t(b) > \ell_t(v_1)$ *and* $X_t = b$ can simultaneously occur at most $\sqrt{T}$ times, because b is decreased by $T^{-1/2}$ when this happens. We can exploit this observation as follows: when Exp3 plays b and does not sell, then we feed the algorithm a reduced loss of zero. This has the effect of underestimating the algorithm loss by an amount which is bounded by the number of times the algorithm got a reduced loss. This effect is only increasing the regret by at most $\sqrt{T}$, since is the largest number of times b can be decreased while being larger than v_1.

More formally, our algorithm runs Exp3 on the two prices b_t and v_2, where b_t is dynamically adjusted during the execution. Price b_t is used to locate v_1 and is initially set to v_2. Exp3 is run with reduced losses $\widetilde{\ell}_t$ of the form $\widetilde{\ell}_t(b_t) = \ell_t(b_t)\mathbb{I}\{V_t \ge b_t\}$ and $\widetilde{\ell}_t(v_2) = \ell_t(v_2)$, where $\ell_t(x) = 1 - x\,\mathbb{I}\{V_t \ge x\}$ are the true losses. Note that, whenever $b_t > v_1$, $V_t = v_1$ implies $\widetilde{\ell}_t(b_t) = 0 = \ell_t(v_1)$, and $V_t = v_2$ implies $\widetilde{\ell}_t(b) \le \ell(v_1)$. Since b_t is random (it depends on $X_1, \ldots, X_{t-1}$), $\widetilde{\ell}_t(b_t)$ and $\ell_t(b_t)$ are also random. Technically, this corresponds to running Exp3 with a nonoblivious adversary —see, e.g., [23, Remark 4.1]. However, the regret bounds of Exp3 hold unchanged even for nonoblivious adversaries.

During the execution of Exp3, b_t is adjusted according to the following rule: if b_t is posted at time t and $V_t < b_t$, then $b_{t+1} = b_t - T^{-1/2}$. For all t let $V_t \in \{v_1, v_2\}$ be the value played by the adversary at time t and $X_t \in \{b_t, v_2\}$ be the price posted by the algorithm.

Theorem 1.10. *The regret of the above algorithm satisfies* $R_T \le 2\sqrt{T} + \sqrt{4T\ln 2}$.

Proof. Since $b_1 \le v_2 < 1$, and because $X_t = b_t$ and $\mathbb{I}\{V_t < b_t\}$ imply $b_{t+1} = b_t - T^{-1/2}$, we have that

$$\sum_{t=1}^{T}\mathbb{I}\{X_t = b_t\}\,\mathbb{I}\{V_t < b_t\} < \sqrt{T}\ .$$

Therefore, the true total loss of Exp3 deterministically relates to its reduced loss as follows,

$$\sum_{t=1}^{T}\widetilde{\ell}_t(X_t) = \sum_{t=1}^{T}\ell_t(X_t) - \sum_{t=1}^{T}\ell_t(b_t)\mathbb{I}\{X_t = b_t\}\,\mathbb{I}\{V_t < b_t\} \ge \sum_{t=1}^{T}\ell_t(X_t) - \sqrt{T}\ . \tag{1.15}$$

If $b_t \le v_1$ for some t, then b_t is always accepted. Hence it is never decreased further, which in turn implies that $b_t > v_1 - T^{-1/2}$ holds for all t. So we have that $\widetilde{\ell}_t(b_t) \le \ell_t(v_1) + T^{-1/2}$. Recalling that $\widetilde{\ell}_t(v_2) = \ell_t(v_2)$ and using Exp3 regret bound (see, e.g., [18, Theorem 3.1]) applied to the nonoblivious reduced losses $\widetilde{\ell}_t$, we obtain

$$\mathbb{E}\left[\sum_{t=1}^{T}\ell_t(X_t)\right] - \sqrt{T} \le \mathbb{E}\left[\sum_{t=1}^{T}\widetilde{\ell}_t(X_t)\right] \qquad \text{(using (1.15))}$$

$$\le \min\left\{\sum_{t=1}^{T}\widetilde{\ell}_t(b_t), \sum_{t=1}^{T}\widetilde{\ell}_t(v_2)\right\} + \sqrt{4T\ln 2} \le \min\left\{\sum_{t=1}^{T}\Bigl(\ell_t(v_1) + T^{-1/2}\Bigr), \sum_{t=1}^{T}\ell_t(v_2)\right\} + \sqrt{4T\ln 2}\ .$$

Therefore, we get

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(X_t)\right] \leq \min_{v \in \{v_1, v_2\}} \sum_{t=1}^{T} \ell_t(v) + 2\sqrt{T} + \sqrt{4T \ln 2}$$

concluding the proof. □

Chapter 2

Cooperative Online Learning

We study an asynchronous online learning setting with a network of agents. At each time step, some of the agents are activated, requested to make a prediction, and pay the corresponding loss. The loss function is then revealed to these agents and also to their neighbors in the network. Our results characterize how much knowing the network structure affects the regret as a function of the model of agent activations. When activations are stochastic, the optimal regret (up to constant factors) is shown to be of order $\sqrt{\alpha T}$, where T is the horizon and α is the independence number of the network. We prove that the upper bound is achieved even when agents have no information about the network structure. When activations are adversarial the situation changes dramatically: if agents ignore the network structure, a $\Omega(T)$ lower bound on the regret can be proven, showing that learning is impossible. However, when agents can choose to ignore some of their neighbors based on the knowledge of the network structure, we prove a $\mathcal{O}(\sqrt{\overline{\chi} T})$ sublinear regret bound, where $\overline{\chi} \geq \alpha$ is the clique-covering number of the network.

2.1 Introduction

Distributed asynchronous online learning settings with communication constraints arise naturally in several applications. For example, large-scale learning systems are often geographically distributed, and in domains such as finance or online advertising, each agent must serve high volumes of prediction requests. If agents keep updating their local models in an online fashion, then bandwidth and computational constraints may preclude a central processor from having access to all the observations from all sessions, and synchronizing all local models at the same time. An example in a different domain is mobile sensor networks cooperating towards a common goal, such as environmental monitoring. Sensor readings provide instantaneous, full-information feedback and energy-saving constraints favor short-range communication over long-range. Online learning algorithms distributed over spatial locations have already been proposed for problems in the field of climate informatics [62, 63], and have shown empirical performance advantages compared to their global (i.e., non-spatially distributed) online learning counterparts.

Motivated by these real-life applications, we introduce and analyze an online learning setting in which a network of agents solves a common online convex optimization problem by sharing feedback with their network neighbors. Agents do not have to be synchronized. At each time step, only some of the agents are requested to make a prediction and pay the corresponding loss: we call these agents "active". Because the feedback (i.e., the current loss function) received by the active agents is communicated to their neighbors, both active agents and their neighbors can use the feedback to update their local models. The lack of global synchronization implies that agents who are not requested to make a prediction get "free feedback" whenever someone is active in their neighborhood. Since in online convex optimization the sequence of loss functions is fully arbitrary, it is not clear whether this free feedback can improve the system's performance. In this chapter, we characterize under which conditions and to what extent such improvements are possible.

Our goal is to control the network regret, which we define by summing the average instantaneous regret of the active agents at each time step. In order to build some intuition on this problem, consider the following two extreme cases where, for the sake of simplicity, we assume exactly one agent is active at each time step. If no communication is possible among the agents, then each agent v learns in isolation over the subset T_v of time steps when they are active. Assuming each agent runs a standard online learning algorithm with regret bounded by $\mathcal{O}(\sqrt{T})$ —such as Online Mirror Descent (OMD)— the network regret is at most of order $\sum_v \sqrt{T_v} \leq \sqrt{NT}$, where $T = \sum_v T_v$ and N is the number of agents. Next, consider a fully connected graph, where agents share their feedback with the rest of the network. Each local instance of OMD now sees the same loss sequence as the other instances, so the sequence of predictions is the same, no matter which agents are chosen to be active. The network regret is then bounded by $\mathcal{O}(\sqrt{T})$, as in the single-instance case. Our goal is to understand the regret when the communication network corresponds to an arbitrary graph G.

We consider two natural activation mechanisms for the agents: stochastic and adversarial. In the stochastic setting, at each time step t each agent v is independently active with probability q_v, where q_v is a fixed and unknown number in $[0, 1]$. Under this assumption, we show that when each agent runs OMD, the network regret is $\mathcal{O}(\sqrt{\alpha T})$, where $\alpha \leq N$ is the independence number of the communication graph. Note that this bound smoothly interpolates the two extreme cases of no communication ($\alpha = N$) and full communication ($\alpha = 1$). From this viewpoint, α can be viewed as the number of "effective instances" that are implicitly maintained by the system. It is not hard to prove that this upper bound cannot be improved upon: fix a network G and a maximal independent set in G of size α. Define $q_v = 1/\alpha$ if v belongs to the independent set and 0 otherwise. Then no two nodes that can ever become active are adjacent in G, and we reduced the problem to that of learning with α non-commmunicating agents over T/α time steps. Since there are instances of the standard online convex optimization problem on which any agent strategy has regret $\Omega(\sqrt{T})$, we obtain that the network regret must be at least of order $\alpha\sqrt{T/\alpha} = \sqrt{\alpha T}$. Note that this lower bound also applies to algorithms that have complete preliminary knowledge of the graph structure, and can choose to ignore or process any feedback coming from their neighbors. In contrast, the OMD instances used to prove the upper bound are fully oblivious both to the graph structure and to the source of their feedback (i.e., whether their agent is active as opposed to being the neighbor of an active agent).

In the adversarial activation setting, nodes are activated according to some unknown deterministic schedule. Surprisingly, under the same assumption of obliviousness about the feedback source which we used to prove the $\mathcal{O}(\sqrt{\alpha T})$ upper bound for stochastic activations, we show that on certain network topologies a deterministic schedule of activations can force a *linear* regret on *any* algorithm, thus making learning impossible. On the other hand, if agents are free to use feedback only from a subset of their neighbors chosen with knowledge of the graph structure, then the network regret of OMD is $\mathcal{O}(\sqrt{\overline{\chi} T})$, where $\overline{\chi}$ is the clique-covering number of the communication graph. Hence, unlike the stochastic case, where the knowledge of the graph is not required to achieve optimality, in the adversarial case the ability of choosing the feedback source based on the graph structure is both a necessary and sufficient condition for sublinear regret.

The extension of the OMD analysis to a multiagent setting with communication (Theorem 2.2), and the lower bound for the adversarial activation setting (Theorem 2.5) are the main technical novelties of the chapter.

2.2 Related works

The study of cooperative nonstochastic online learning on networks was pioneered in [6], where they investigated a bandit setting in which the communication graph is a clique, users are clustered so that the loss function at time t may differ across clusters, and some users may be non-cooperative. More recently, a similar line of work was pursued in [24], where they derive graph-dependent regret bounds for nonstochastic bandits on arbitrary networks when the loss function is the same for all nodes and the feedbacks are broadcast to the network with a delay corresponding to the shortest path distance on the graph. Although their regret bounds —like ours— are expressed in terms of the network independence number, this happens for reasons that are very different from ours, and by means of a different analysis. In their setting all agents are simultaneously active at each time step, and sharing the feedback serves the purpose of reducing the variance of the importance-weighted loss estimates. A node with many neighbors observes the current loss function evaluated at all the points corresponding to actions played by the neighbors. Hence, in that context cooperation serves to bring the bandit feedback closer to a full information setting.

In contrast, we study a full information setting in which agents get free and meaningful feedback only when they are not requested to predict.[1] Therefore, in our setting cooperation corresponds to faster learning (through the free feedback that is provided over time) *within the full information model*, as opposed to [24] where cooperation *increases feedback within a single time-step*. An even more recent work considering bandit networks is [58]. They study a stochastic bandit model with simultaneous activation and constraints on the amount of communication between neighbors. Their regret bounds scale with the spectral gap of the communication network. The work [79] investigates a different partial information model of prediction with expert advice where each agent is paired with an expert, and agents see only the loss of their own expert. The communication model includes delays, and the regret bound depends on a quantity related to the mixing time of a certain random walk on the network. The authors of [97] study a decentralized online learning setting in which losses are characterized by two components, one adversarial and another stochastic. They show upper bounds on the regret in terms of a constant representing the magnitude of the adversarial component and another constant measuring the randomness of the stochastic part.

The idea of varying the amount of feedback available to learning agents has also appeared in single-agent settings. In the *sleeping experts* model [35], different subsets of actions are available to the learner at different time steps. In our multi-agent setting, instead, actions are always available while the agents are occasionally sleeping. An algorithmic reduction between the two settings seems unlikely to exist because actions and agents play completely different roles in the learning process. In the *learning with feedback graphs* model [2, 57], each selection of an action reveals to the learner the loss of the actions that are adjacent to it in a given graph. In our model, each time an active agent plays an action, the loss vector is revealed to the agents that are adjacent to the active learner. There is again a similarity between actions and agents in the two settings, but to the best of our knowledge there is no algorithmic reduction from multi-agent problems to single-agent problems. Yet, it should not come as a surprise that some general graph-theoretic tools —like Lemma 2.2 — are used in the analysis of both single-agent and multi-agent models.

A very active area of research involves distributed extensions of online convex optimization, in which the global loss function is defined as a sum of local convex functions, each associated with an agent. Agents are run over the local optimization problem corresponding to their local functions and communicate with their neighborhood to find a point in the decision set approximating the loss of the best global action. This problem has been studied in various settings: distributed convex optimization —see, e.g., [32, 80] and references therein, distributed online convex optimization [45], and a dynamic regret extension of distributed online convex optimization [81]. Unlike our work, these papers consider distributed extensions of OMD (and Nesterov dual averaging) based on generalizations of the consensus problems. The resulting performance bounds scale inversely in the spectral gap of the communication network.

2.3 Preliminaries and definitions

Let $G = (V, E)$ be a communication network, i.e., an undirected graph over a set V of N *agents*. Without loss of generality, assume $V = \{1, \dots, N\}$. For any agent $v \in V$, we denote by $\mathcal{N}_v$ the set of nodes containing the agent v and the neighborhood $\big\{w \in V \mid (v, w) \in E\big\}$. The *independence number* α_G is the cardinality of the biggest *independent set* of G, i.e., the cardinality of the biggest subset of agents, no two of which are neighbors.

We study the following cooperative online convex optimization protocol: initially, hidden from the agents, the environment picks a sequence of subsets $S_1, S_2, \ldots \subseteq V$ of *active* agents and a sequence of differentiable convex real loss functions $\ell_1, \ell_2, \ldots$ defined on a convex decision set $\mathcal{X} \subset \mathbb{R}^d$. Then, for each time step $t \in \{1, 2, \ldots\}$,

1. each agent $v \in S_t$ predicts with $\boldsymbol{x}_t(v) \in \mathcal{X}$,
2. each agent $v \in \bigcup_{v \in S_t} \mathcal{N}_v$ receives ℓ_t as feedback,
3. the system incurs the loss $\frac{1}{|S_t|} \sum_{v \in S_t} \ell_t\big(\boldsymbol{x}_t(v)\big)$ (defined as 0 when $S_t \equiv \varnothing$).

[1]Two adjacent agents that are simultaneously active exchange their feedback, but this does not bring any new information to either agent because we are in a full information setting and the loss function is the same for all nodes.

We assume each agent v runs an instance of the same online algorithm. Each instance learns a local model generating predictions $\boldsymbol{x}_t(v)$. This local model is updated whenever a feedback ℓ_t is received. We call *paid feedback* the feedback ℓ_t received by v when $v \in S_t$ (i.e., the agent is active) and *free feedback* the feedback ℓ_t received by v when $v \in \left(\bigcup_{v \in S_t} \mathcal{N}_v\right) \setminus \{S_t\}$ (i.e., the agent is not active but in the neighborhood of some active agent). The goal is to minimize the *network regret* as a function of the unknown number T of time steps,

$$R_T = \sum_{t=1}^{T} \frac{1}{|S_t|} \sum_{v \in S_t} \ell_t\big(\boldsymbol{x}_t(v)\big) - \inf_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} \ell_t(\boldsymbol{x}) \tag{2.1}$$

Note that only the losses of active agents contribute to the network regret.

2.4 Online Mirror Descent

Algorithm 9: Online Mirror Descent

```
  Input: $\sigma_t$-strongly convex regularizers $g_t \colon \mathcal{X} \to \mathbb{R}$ for $t \in \{1, 2, \dots\}$.
  Initialization: set $\boldsymbol{\theta}_1 = \boldsymbol{0} \in \mathbb{R}^d$.
1 for $t \in \{1, 2, \dots\}$ do
2     choose $\boldsymbol{w}_t = \nabla g_t^*(\boldsymbol{\theta}_t)$;
3     observe $\nabla \ell_t(\boldsymbol{w}_t) \in \mathbb{R}^d$;
4     update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \nabla \ell_t(\boldsymbol{w}_t)$;
```

We now review the standard Online Mirror Descent algorithm (OMD) —see Algorithm 9— and its analysis. Let $f \colon \mathcal{X} \to \mathbb{R}$ be a convex function. We say that $f^* \colon \mathbb{R}^d \to \mathbb{R}$ is the *convex conjugate* of f if $f^*(\boldsymbol{x}) = \sup_{\boldsymbol{w} \in \mathcal{X}} \big(\boldsymbol{x} \cdot \boldsymbol{w} - f(\boldsymbol{w})\big)$. We say that f is *σ-strongly convex* on $\mathcal{X}$ with respect to a norm $\|\cdot\|$ if there exists $\sigma \geq 0$ such that, for all $\boldsymbol{u}, \boldsymbol{w} \in \mathcal{X}$ we have $f(\boldsymbol{u}) \geq f(\boldsymbol{w}) + \nabla f(\boldsymbol{w}) \cdot (\boldsymbol{u} - \boldsymbol{w}) + \frac{\sigma}{2} \|\boldsymbol{u} - \boldsymbol{w}\|^2$. The following well-known result can be found in [83, Lemma 2.19 and subsequent paragraph].

Lemma 2.1. *Let $f \colon \mathcal{X} \to \mathbb{R}$ be a strongly convex function on $\mathcal{X}$. Then the convex conjugate f^* is everywhere differentiable on $\mathbb{R}^d$.*

The following result —see, e.g., [70, bound (6) in Corollary 1 with F set to zero]— shows an upper bound on the regret of OMD.

Theorem 2.1. *Let $g \colon \mathcal{X} \to \mathbb{R}$ be a differentiable function σ-strongly convex with respect to $\|\cdot\|$. Then the regret of OMD run with $g_t = \frac{\sqrt{t}}{\eta} g$, for $\eta > 0$, satisfies*

$$\sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t) - \inf_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} \ell_t(\boldsymbol{x}) \leq \frac{D}{\eta} \sqrt{T} + \frac{\eta}{2\sigma} \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \|\nabla \ell_t\|_*^2$$

where $D = \sup g$ and $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. If $\sup \|\nabla \ell_t\|_* \leq L$, then choosing $\eta = \sqrt{2\sigma D}/L$ gives $R_T \leq L\sqrt{2DT/\sigma}$.

A popular instance of OMD is the standard online gradient descent algorithm, corresponding to choosing $\mathcal{X}$ equal to a closed Euclidean ball centered at the origin, and setting $g = \frac{1}{2} \|\cdot\|^2$ for all t, where $\|\cdot\|$ is the Euclidean norm. Another instance is the Hedge algorithm for prediction with expert advice, corresponding to choosing $\mathcal{X}$ equal to the probability simplex, and setting $g(\boldsymbol{p}) = \sum_i p_i \ln p_i$.

2.5 Stochastic activations

In this section we analyze the performance of OMD when the sets S_t of active agents are chosen stochastically. As discussed in the introduction, in this setting we do not require any ad-hoc interface between each OMD instance and the rest of the network. In particular, we make the following assumption.

Assumption 2.1 (Oblivious network interface)**.** *An online algorithm A is run with an oblivious network interface if for each agent v it holds that:*

1. *v runs an instance A_v of A,*
2. *A_v uses the same initialization and learning rate as the other instances,*
3. *A_v makes predictions and updates while being oblivious to whether $v \in S_t$ or $v \in \left(\bigcup_{v\in S_t} \mathcal{N}_v\right) \setminus \{S_t\}$.*

This assumption implies that each instance is oblivious to both the network topology and the location of the agent in the network. Moreover, instances make an update whenever they have the opportunity to do so, (i.e., whenever they or some of their neighbors are active). The purpose of this assumption is to show that communication might help OMD even without any network-specific tuning. In concrete applications, one might use ad-hoc OMD variants that rely on the knowledge of the task at hand, and decrease the regret even further. However, the lower bound proven in Section 1.4 shows that the regret cannot be decreased significantly even when agents have full knowledge of the graph.

We start by considering a slightly simplified stochastic activation setting, where only a single agent is activated at each time step (i.e., $|S_t| = 1$ for all t). The more general stochastic case is analyzed at the end of this section. We assume that the active agents $v_1, v_2, \dots$ are drawn i.i.d. from an unknown fixed distribution q on V. The main result of this section is an upper bound on the regret of the network when all agents run the basic OMD (Algorithm 9) with an oblivious network interface. We show that in this case the network achieves the same regret guarantee as the single-agent OMD (Theorem 2.1) multiplied by the square root of independence number of the communication network.

Before proving the main result, we state a combinatorial lemma that allows to upper bound the sum of a ratio of probabilities over the vertices of an undirected graph with the independence number of the graph [38, 57]. The proof is included for completeness.

Lemma 2.2. *Let $G = (V, E)$ be an undirected graph with independence number α_G and q any probability distribution on V such that $Q_v = \sum_{w\in\mathcal{N}_v} q_v > 0$ for all $v \in V$. Then*

$$\sum_{v\in V} \frac{q_v}{Q_v} \le \alpha_G$$

Proof. Initialize $V_1 = V$, fix $w_1 \in \arg\min_{w\in V_1} Q_w$, and denote $V_2 = V \setminus \mathcal{N}_{w_1}$. For $k \ge 2$ fix $w_k \in \arg\min_{w\in V_k} Q_w$ and shrink $V_{k+1} = V_k \setminus \mathcal{N}_{w_k}$ until $V_{k+1} = \varnothing$. Being G undirected $w_k, wehave \notin \bigcup_{s=1}^{k-1} \mathcal{N}_{w_s}$, therefore the number m of times that an action can be picked this way is upper bounded by α_G. Denoting $\mathcal{N}'_{w_k} = V_k \cap \mathcal{N}_{w_k}$, this implies

$$\sum_{v\in V} \frac{q_v}{Q_v} = \sum_{k=1}^{m} \sum_{v\in\mathcal{N}'_{w_k}} \frac{q_v}{Q_v} \le \sum_{k=1}^{m} \sum_{v\in\mathcal{N}'_{w_k}} \frac{q_v}{Q_{w_k}} \le \sum_{k=1}^{m} \frac{\sum_{v\in\mathcal{N}_{w_k}} q_v}{Q_{w_k}} = m \le \alpha_G$$

□

The following holds for any differentiable function $g\colon \mathcal{X} \to \mathbb{R}$, σ-strongly convex with respect to some norm $\|\cdot\|$.

Theorem 2.2. *Consider a network $G = (V, E)$ of N agents and assume $S_t = \{v_t\}$ for each t, where v_t is drawn i.i.d. from some fixed and unknown distribution on V. If all agents run OMD with an oblivious network interface and using $g_t = \frac{\sqrt{t}}{\eta} g$, for $\eta > 0$, then the network regret satisfies*

$$\mathbb{E}[R_T] \le \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{\alpha_G T}$$

where $D \geq \sup g$, $L \geq \sup \|\nabla \ell_t\|_*$, and $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. In particular, choosing $\eta = \sqrt{2\sigma D}/L$ gives $\mathbb{E}[R_T] \leq L\sqrt{2D\alpha_G T/\sigma}$.

Proof. Fix $\boldsymbol{x} \in \mathcal{X}$, any sequence of realizations $v_1, \ldots, v_T$, and any v in the support $V' \subset V$ of the activation distribution q. Note that the OMD instance run by v, makes an update at time t only when $v \in \mathcal{N}_{v_t}$. Hence, letting $r_t(v) = \ell_t\big(\boldsymbol{x}_t(v)\big) - \ell_t(\boldsymbol{x})$ and applying Theorem 2.1,

$$\sum_{t=1}^{T} r_t(v)\mathbb{I}\{v \in \mathcal{N}_{v_t}\} \leq \frac{D}{\eta}\sqrt{T_v} + \frac{\eta L^2}{2\sigma}\sum_{t=1}^{T}\frac{\mathbb{I}\{v \in \mathcal{N}_{v_t}\}}{\sqrt{\sum_{s=1}^{t}\mathbb{I}\{v \in \mathcal{N}_{v_s}\}}} \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{T_v} \tag{2.2}$$

where $T_v = \sum_{t=1}^{T}\mathbb{I}\{v \in \mathcal{N}_{v_t}\}$, the addends after the first inequality are intended to be null when the denominator is zero, and we used $\sum_{s=1}^{T_v} s^{-1/2} \leq 2\sqrt{T_v}$. Note that $r_t(v)$ is independent of v_t, as it only depends on the subset of v_s, $s \in \{1, \ldots, t-1\}$, such that $v \in \mathcal{N}_{v_s}$. Denote by Q_v the probability $\mathbb{P}(v \in \mathcal{N}_{v_t}) = \sum_{w \in \mathcal{N}_v} q(w) > 0$. Let $\mathcal{F}_{t-1}$ be the σ-algebra generated by $\{v_1, \ldots, v_{t-1}\}$. Since Q_v is independent of t, $\mathbb{P}\big(v \in \mathcal{N}_{v_t} \mid \mathcal{F}_{t-1}\big) = Q_v$. Therefore, taking expectation with respect to $v_1, \ldots, v_T$ on both sides of (2.2), using $\mathbb{E}[T_v] = Q_v T$, and applying Jensen's inequality, yields

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t(v) Q_v\right] \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{Q_v T} \tag{2.3}$$

Now, letting $R_T(\boldsymbol{x}) = \sum_{t=1}^{T} r_t(v_t)$, we have that $\mathbb{E}\big[R_T(\boldsymbol{x})\big]$ is equal to

$$\mathbb{E}\left[\sum_{v \in V'}\sum_{t=1}^{T} r_t(v)\mathbb{I}\{v_t = v\}\right] = \mathbb{E}\left[\sum_{v \in V'}\sum_{t=1}^{T} r_t(v)\mathbb{E}\big[\mathbb{I}\{v_t = v\} \mid \mathcal{F}_{t-1}\big]\right] = \sum_{v \in V'} q_v \mathbb{E}\left[\sum_{t=1}^{T} r_t(v)\right]$$

Dividing both sides of (2.3) by $Q_v > 0$, we can write

$$\mathbb{E}\big[R_T(\boldsymbol{x})\big] \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sum_{v \in V'} q_v\sqrt{\frac{T}{Q_v}} \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{T\sum_{v \in V'}\frac{q_v}{Q_v}} \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{\alpha T}$$

where in the last two inequalities we applied Jensen's inequality and Lemma 2.2. Observing that $\mathbb{E}[R_T] = \sup_{\boldsymbol{x} \in \mathcal{X}}\mathbb{E}\big[R_T(\boldsymbol{x})\big]$ and recalling that $\boldsymbol{x}$ was chosen arbitrarily in $\mathcal{X}$ concludes the proof. □

Note that the proof of the previous result gives a tighter upper bound on the network regret in terms of the independence number $\alpha' \leq \alpha$ of the subgraph induced by the support V' of q.

Next, we consider the setting in which we allow the activation of more than one agent per time step. At the beginning of the process, the environment draws an i.i.d. sequence of Bernoulli random variables $X_1(v), X_2(v), \ldots$ with some unknown fixed parameter $q_v \in [0,1]$ for each agent $v \in V$. The active set at time t is then defined as $S_t = \{v \in V \mid X_t(v) = 1\}$. Note that, unlike the previous setting, now $\sum_{v \in V} q_v \neq 1$ in general.

We state an upper bound on the regret that the network incurs if all agents run OMD with an oblivious network interface (for a proof, see Section 2.9). Our upper bound is expressed in terms of a constant depending on the probabilities of activating each agent and such that $Q \leq 1.6(\alpha_G + 1)$. The result holds for any differentiable function $g \colon \mathcal{X} \to \mathbb{R}$, σ-strongly convex with respect to some norm $\|\cdot\|$.

Theorem 2.3. *Consider a network $G = (V, E)$ of N agents. Assume that, at each time step t each agent v is independently activated with probability $q_v \in [0,1]$. If all agents run OMD with an oblivious network interface and using $g_t = \frac{\sqrt{t}}{\eta}g$, for $\eta > 0$, the network regret satisfies*

$$\mathbb{E}[R_T] \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{QT}$$

for some $Q \geq 0$, $D \geq \sup g$, and $L \geq \sup \|\nabla \ell_t\|_*$, where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. In particular, choosing $\eta = \sqrt{2\sigma D}/L$ gives $\mathbb{E}[R_T] \leq L\sqrt{2DQT/\sigma}$.

In order to compare the previous upper bound to Theorem 2.2, consider the case $q_v = q$ for all $v \in V$. Without loss of generality, assume $q > 0$ (the regret is zero when q vanishes). Then

$$Q = Q(q) = \frac{1}{N} \sum_{v \in V} \frac{1 - (1-q)^N}{1 - (1-q)^{|\mathcal{N}_v|}}$$

(for a proof, see Theorem 2.3 in Section 2.9 and proceed as in the proof of Lemma 2.2). A direct computation of the sign of the first derivative of the addends $q \mapsto \frac{1-(1-q)^N}{1-(1-q)^{|\mathcal{N}_v|}}$ shows that these functions are decreasing in q, hence $1 = \lim_{q \to 1^-} Q(q) \le Q \le \lim_{q \to 0^+} Q(q) = \sum_{v \in V} \frac{1}{|\mathcal{N}_v|} \le \alpha_G$ where the last inequality follows by Lemma 2.2. Note that the lower bound $Q \ge 1$ is attained if the probabilities of picking agents at each time step are all 1. In this case all agents are activated at each time step, the graph structure over the set of agents becomes irrelevant and the model reduces to a single-agent problem. The inequality $Q(q) \le \alpha_G$ is not a coincidence due to the constant q. Indeed, one can prove that this is always the case, up to a small constant factor (for a proof., see Lemma 2.3 in Section 2.9).

The previous results shows that paying the average price of multiple activations is never worse (up to a small constant factor) than drawing a single agent per time step, and it can be significantly better. A similar argument shows a tighter bound $Q \le \max\{3, \alpha_G\}$ when the activation probabilities satisfy $\sum_{v \in V} q_v = 1$, which allows to recover the upper bound on the network regret proven in Theorem 2.2. This is consistent with the intuition that —in expectation— picking a single agent at random according to a distribution $q = (q_1, \dots, q_N)$ is the same as picking each v independently with probability q_v. Similarly to the case $|S_t| = 1$, the previous result gives a tighter upper bound on the network regret in terms of the independence number $\alpha' \le \alpha$ of the subgraph induced by the subset V' of V containing all agents v with $q_v > 0$. Note that the setting discussed in this section smoothly interpolates between the single-agent setting ($q_v = 1$ for all v), cooperative learning with one agent stochastically activated at each time step ($\sum_v q_v = 1$), and beyond ($\sum_v q_v < 1$), where a non trivial fraction of the total number rounds is skipped.

2.6 Lower bound for stochastic activations

In this section we show that, for any communication network G with stochastic agent activations, the best possible regret rate is of order $\Omega\big(\sqrt{\alpha_G T}\big)$. This holds even when agents are not restricted to use an oblivious network interface. The idea is that if the distribution from which active agents are drawn is supported on an independent set of cardinality α_G, then the problem reduces to that of an edgeless graph with α_G agents. We sketch the proof for the case when $|S_t| = 1$.

Theorem 2.4. *There exists a convex decision set in $\mathbb{R}^d$ such that, for each communication network G and for arbitrary (and possibly different) online learning algorithms run by the agents, $\mathbb{E}[R_T] = \Omega\big(\sqrt{\alpha T}\big)$ for some sequence $(S_1, \ell_1), \dots, (S_T, \ell_T)$, where $S_t = \{v_t\}$, v_t is drawn i.i.d. from some fixed distribution on V, and the expectation is taken with respect to the random draw of the $v_1, \dots, v_T$.*

Proof sketch. Let $\mathcal{X}$ be the probability simplex in $\mathbb{R}^d$. Let $G = (V, E)$ be any communication graph and α its independence number. We consider linear losses defined on $\mathcal{X}$. Let q be the uniform distribution over a maximal independent set $A = \{a_1, \dots, a_\alpha\} \subset V$. Fix now any cooperative online linear optimization algorithm for this setting. Since each active agent v_t belongs to A for all $t \in \{1, \dots, T\}$ with probability 1, it suffices to analyze the updates of the algorithm for these agents. Indeed, no other agent incurs any loss at any time-step. Since A is an independent set, each agent a_i makes an update at round t if and only if $v_t = a_i$. This happens with probability $q(a_i) = 1/\alpha$, independently of t. Each agent a_i is therefore running an independent single-agent online linear optimization problem for an average of T/α rounds. It is well-known [42, Theorem 3.2] that any algorithm for online linear optimization on the simplex with losses bounded in $[0, 1]$ incurs $\Omega\big(\sqrt{T/\alpha}\big)$ regret over T/α rounds in the worst case. Consequently, the regret of the network satisfies $R_T = \Omega\big(\alpha\sqrt{T/\alpha}\big) = \Omega\big(\sqrt{\alpha T}\big)$. $\square$

An analogous lower bound can be proven for the case of multiple agent activations per time step. Indeed, define $q_v = 1/\alpha$ for each agent v belonging to some fixed maximal independent set and $q_v = 0$ otherwise.

This again leads to α independent single-agent online linear optimization problems for an average of T/α rounds each, and an argument similar to the one in the proof of Theorem 2.4 gives the result.

2.7 Nonstochastic activations

In this section we drop the stochasticity assumption on the agents' activations and focus on the case where active agents are picked from V by an adversary. The goal is to control the regret (2.1) for *any* individual sequence of pairs $(\ell_1, S_1), (\ell_2, S_2), \dots$ where ℓ_t is a convex loss and $S_t \subseteq V$, without any stochastic assumptions on the mechanism generating these pairs. For the rest of this section, we focus on the special case where $|S_t| = 1$ for all t and denote by v_t the active node at time t.

We start by proving that learning with adversarial activations is impossible if we use an oblivious network interface. We prove this result in the setting of prediction with expert advice with two actions and binary losses, a special case of online convex optimization. The idea of the lower bound is that if the communication network is a star graph, the environment is able to make both actions look equally good to all peripheral agents, even if one of the two actions is actually better than the other. This is done by drawing the good action at random, then activating an agent depending on the outcome of the draw. For a small fraction of the times the good action has loss one, the central agent is activated. Since the central agent shares feedback with all peripheral agents, we can amplify this loss by a factor of N, and thus make the good action look to all peripheral agents as bad as the bad action.

Theorem 2.5. *For each $N > 3$ there exists a convex decision set in $\mathbb{R}^2$ and a graph G with N vertices such that, whenever N agents are run on G using instances of any online learning algorithm with an oblivious network interface, then $R_T = \Omega(T)$ for some sequence $(\ell_1, v_1), \dots, (\ell_T, v_T)$ of convex losses and active agents.*

Proof. Fix $N > 3$ and let $\mathcal{X}$ be the probability simplex in $\mathbb{R}^2$. Let $G = (V, E)$ be the star graph with central agent a_0, and peripheral agents $a_1, \dots, a_{N-1}$. Because our losses are linear on $\mathcal{X}$, the online convex optimization problem is equivalent to prediction with expert advice with two experts (or actions), and we may denote losses using loss vectors $\ell_t = \big(\ell_t(1), \ell_t(2)\big)$ where 1 and 2 index the actions. A *good action* $J \in \{1, 2\}$ is drawn uniformly at random. Denote the other one (i.e., the *bad* one) by J_{B}. To keep notation tidy, we define loss vectors by $\ell_t = \big(\ell_t(J), \ell_t(J_{\mathrm{B}})\big)$. Fix any $\varepsilon \in \big(0, \frac{N-1}{2(N-2)}\big)$. The loss vectors ℓ_t are drawn i.i.d. at random, according to the following joint distribution:

$$\mathbb{P}\big(\ell_t = (0,1)\big) = \frac{1}{2} \qquad \mathbb{P}\big(\ell_t = (1,0)\big) = \frac{1}{2} - \varepsilon + \frac{\varepsilon}{N-1} \qquad \mathbb{P}\big(\ell_t = (0,0)\big) = \varepsilon - \frac{\varepsilon}{N-1}$$

Recall that only a single agent v_t is active at any time. At each time step t, the adversary decides whether to activate the central agent a_0 or a peripheral agent, depending on the realization of ℓ_t. If $\ell_t(J) = 0$, then a random peripheral agent is activated. Otherwise, we set

$$\mathbb{P}\big(\ell_t = (1,0),\, v_t = a_0\big) = \frac{\varepsilon}{N-1} \quad \text{and} \quad \mathbb{P}\big(\ell_t = (1,0),\, v_t = a_i\big) = \frac{1/2 - \varepsilon}{N-1} \quad a_1, \dots, a_{N-1}$$

Note that when $v_t = a_0$, then all peripheral agents receive feedback ℓ_t. Similarly, when a peripheral agent is active at time t, then a_0 receives feedback ℓ_t. For $b_1, b_2 \in \{0, 1\}$, let $E(a_i, b_1, b_2)$ be the event: agent a_i receives the loss vector $\ell_t = (b_1, b_2)$ as feedback. The following statements then hold for each peripheral agent a_i,

$$\mathbb{P}\big(E(a_i,0,1)\big) = \frac{1/2}{N-1} \qquad \mathbb{P}\big(E(a_i,0,0)\big) = \frac{\varepsilon}{N-1} - \frac{\varepsilon}{(N-1)^2}$$
$$\mathbb{P}\big(E(a_i,1,0)\big) = \frac{1/2 - \varepsilon}{N-1} + \frac{\varepsilon}{N-1} = \frac{1/2}{N-1}$$

Hence, each instance managed by a peripheral agent observes loss vectors $(1,0)$ and $(0,1)$ with the same probability proportional to $1/2$, and loss vector $(0,0)$ with probability proportional to $\varepsilon(N-1)/(N-2)$. Since the network interface is oblivious, the instance cannot distinguish between paid and free feedback (which would reveal the good action), and incurs an expected loss of $1/2$ each time $\ell_t \in \{(0,1),(1,0)\}$. Using the fact that a peripheral agent is active when $\ell_t \in \{(0,1),(1,0)\}$ with probability $1/2 + 1/2 - \varepsilon = 1 - \varepsilon$, the system's expected total loss is at least $\frac{1-\varepsilon}{2}T$ (we lower bound the loss of the central agent by zero). Since the expected loss of J is $\left(1/2 - \varepsilon + \frac{\varepsilon}{N-1}\right)T$, the expected regret of the system satisfies

$$\mathbb{E}[R_T] \geq \left(\frac{1-\varepsilon}{2} - \frac{1}{2} + \varepsilon - \frac{\varepsilon}{N-1}\right) T \geq \frac{T}{8}$$

where we picked $\varepsilon = (N-1)/(N-2)$ and used $(N-3)/(N-2) \geq 1/2$ in the last inequality. Therefore, there exists some sequence $(\ell_1, S_1), \dots, (\ell_T, S_T)$ such that $R_T \geq T/8$, concluding the proof. □

We complement the above negative result by showing that when algorithms are run without the oblivious network interface, and agents are free to use feedback only from a subset of their neighbors chosen with knowledge of the graph structure, then the network regret of OMD is $\mathcal{O}(\sqrt{\overline{\chi}_G T})$. The quantity $\overline{\chi}_G$ is the clique-covering number of the communication graph G, which corresponds to the smallest cardinality of a clique cover of G (a clique cover is a partition of the vertices such that the nodes in every element of the partition form a clique in the graph). The intuition behind this result is simple: fix a clique cover and let the agents in the same clique of the cover know each other. Now, if each agent ignores all feedback coming from agents in other cliques, then the agents in the same clique make exactly the same sequence of prediction and updates. Therefore, the effective number of OMD instances that are being run is equal to $\overline{\chi}_G$.

The following result holds for any differentiable function $g \colon \mathcal{X} \to \mathbb{R}$, σ-strongly convex with respect to some norm $\|\cdot\|$.

Theorem 2.6. *Consider a network $G = (V, E)$ of N agents, a clique cover $\{K_1, \dots, K_M\}$ where $M = \overline{\chi}_G$, and let $K(v)$ be the unique element of the cover which each $v \in V$ belongs to. For any sequence $v_1, v_2, \dots \in V$ of active agents, assume each agent $v \in V$ runs OMD using $g_t = \frac{\sqrt{t}}{\eta} g$ (with $\eta > 0$) while making updates only at those time steps t such that $v_t \in K(v)$. Then the network regret satisfies*

$$\mathbb{E}[R_T] \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right) \sqrt{\overline{\chi}_G T}$$

where $D \geq \sup g$, $L \geq \sup \|\nabla \ell_t\|_*$, and $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. In particular, choosing $\eta = \sqrt{2\sigma D}/L$ gives $\mathbb{E}[R_T] \leq L\sqrt{2D\overline{\chi}_G T/\sigma}$.

Proof. Fix any clique K_c and any $v \in K_c$. Let T_c be the time steps such that $v_t \in K_c$. Since each agent $v \in K_c$ ignores the feedback coming from other cliques, the nodes in K_c perform exactly the same updates, and therefore make exactly the same predictions. This means that, for any $t \in T_c$, the predictions in the set $\{\boldsymbol{x}_t(v) \mid v \in K_c\}$ are all equal to the same common value denoted by $\boldsymbol{x}_t(K_c)$. Fix any $\boldsymbol{x} \in \mathcal{X}$ and, for any $t \in T_c$, let $r_t(K_c) = \ell_t(\boldsymbol{x}_t(K_c)) - \ell_t(\boldsymbol{x})$. By Theorem 2.1 we have that

$$\sum_{t \in T_c} r_t(K_c) \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right) \sqrt{T_c}\ .$$

Therefore, recalling that $r_t(v_t) = \ell_t(\boldsymbol{x}_t(v_t)) - \ell_t(\boldsymbol{x})$ and using Jensen's inequality,

$$\sum_{t=1}^{T} r_t(v_t) = \sum_{c=1}^{\overline{\chi}_G} \sum_{t \in T_c} r_t(K_c) \leq \sum_{c=1}^{\overline{\chi}_G} \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right) \sqrt{T_c} \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right) \sqrt{\overline{\chi}_G T}$$

concluding the proof. □

Theorems 2.5 and 2.6 show that with adversarial activations the knowledge of the graph is crucial for learning (e.g., for achieving sublinear regret). Since $\overline{\chi}_G \geq \alpha_G$, it is not clear whether the better rate $\sqrt{\alpha_G T}$ can be proven in the adversarial activation setting when agents do not use the oblivious network interface.

2.8 Conclusions

We introduced a cooperative learning setting in which agents, sitting on the nodes of a communication network, run instances of an online learning algorithm with the common goal of minimizing their regret. In order to investigate how the knowledge of the graph topology affects regret in cooperative online learning under different activation mechanisms, we introduced the notion of oblivious network interface. This prevents agents from doing any network-specific tuning or even accessing their neighborhood structure. When activations are stochastic, we showed that sharing losses among neighbors is enough to guarantee optimal regret rates even with the oblivious network interface. Surprisingly, when activations are adversarial the situation changes completely. There exist problem instances in which any algorithm that runs with the oblivious network interface suffers linear regret. In this case knowing graph structure is not only necessary to perform optimally, but even to have sublinear regret.

Other interesting variants of this settings could be studied in the future. For example, at the beginning of each round, active agents could be allowed to ask the predictions of some of their neighbors, and base their prediction upon it. In this case, we conjecture that the optimal regret rate would scale with the dominating number δ_G of the graph, which is always smaller or equal to the independence number.

2.9 Deferred proofs on stochastic activations: multiple agents

In this section we present all missing results related to the stochastic activation model with multiple activations per time step. Recall that, at the beginning of the process, the environment draws an i.i.d. sequence of Bernoulli random variables $X_1(v), X_2(v), \dots$ with some unknown fixed parameter $q_v \in [0,1]$ for each agent $v \in V$. The active set at time t is then defined as $S_t = \{v \in V \mid X_t(v) = 1\}$. Note that, unlike when only one agent is active at each time step, now $\sum_{v\in V} q_v \neq 1$ in general. Before the main result, we give some definitions and prove a technical combinatorial lemma that is leveraged in the analysis.

Denote by V' the set of all agents $v \in V$ such that $q_v > 0$. For each $v \in V'$, let

$$c_v = \sum_{S\subset\{1,\dots,N\}\setminus\{v\}} \frac{\lambda_{S,v}}{1+|S|} \tag{2.4}$$

where the convex coefficients $\lambda_{S,v}$ are defined by

$$\left(\prod_{w=1}^{N} q_w\right)\left(\prod_{u\in\{1,\dots,N\}\setminus(\{v\}\cup S)} (1-q_u)\right)$$

Let also Q_v be the probability

$$\mathbb{P}\left(v \in \bigcup_{w\in S_t} \mathcal{N}_w\right) = 1 - \prod_{w\in\mathcal{N}_v} \big(1-q_w\big) > 0 \tag{2.5}$$

that agent v is updated at time t —note that Q_v is independent of t.

Lemma 2.2. *Let $X(1), \dots, X(m)$ be independent Bernoulli random variables with strictly positive parameters $q_1, \dots, q_m$ respectively. Then, for all $v \in \{1, \dots, m\}$,*

$$\mathbb{E}\left[\frac{X(v)}{\sum_{w=1}^m X(w)}\right] = q_v c_v$$

where we define $X(v)/\sum_{w=1}^m X(w) = 0$ when $X(v) = 0$.

Proof. Fix any $v \in \{1, \dots, m\}$. Let S_v be the set $\{1, \dots, m\} \setminus \{v\}$ and let $\mathcal{F}_v$ be the σ-algebra generated by $\big\{X(w) \mid w \in S_v\big\}$. Then

$$\mathbb{E}\left[\frac{X(v)}{\sum_{w=1}^m X(w)}\right] = \mathbb{E}\left[\mathbb{E}\left[\frac{X(v)}{\sum_{w=1}^m X(w)}\middle|\mathcal{F}_v\right]\right] = q_v\,\mathbb{E}\left[\frac{1}{1+\sum_{w\in S_v} X(w)}\right]$$

Denote the last expectation by c_v. Since for all $x \neq 0$, $\int_0^\infty e^{-tx}\,\mathrm{d}t = \frac{1}{x}$, Fubini's theorem yields

$$\begin{aligned}
c_v &= \int_0^\infty \mathbb{E}\left[e^{-t\left(1+\sum_{w\in S_v} X(w)\right)}\right] \mathrm{d}t \\
&= \int_0^\infty e^{-t} \prod_{w\in S_v} \mathbb{E}\left[e^{-tX_t(w)}\right] \mathrm{d}t \\
&= \int_0^\infty e^{-t} \prod_{w\in S_v} \left(q_w e^{-t} + 1 - q_w\right) \mathrm{d}t \\
&= \int_0^1 \prod_{w\in S_v} \left(q_w x + 1 - q_w\right) \mathrm{d}x \\
&= \int_0^1 \sum_{S\subset S_v} x^{|S|} \left(\prod_{w\in S} q_w\right) \left(\prod_{u\in S_v\setminus S} (1-q_u)\right) \mathrm{d}x
\end{aligned}$$

Now set $\lambda_{S,v} = \left(\prod_{w\in S} q_w\right)\left(\prod_{S_v\setminus S}(1-q_u)\right)$ and note that $\sum_{S\subset S_v} \lambda_{S,v} = \prod_{w\in S_v}(q_w + 1 - q_w) = 1$. Substituting $\lambda_{S,v}$ in the last identity gives

$$c_v = \sum_{S\subset S_v} \lambda_{S,v} \int_0^1 x^{|S|}\,\mathrm{d}x = \sum_{S\subset S_v} \frac{\lambda_{S,v}}{1+|S|}$$

□

We now give an upper bound on the regret that the network incurs if all agents run OMD with an oblivious network interface. Our upper bound is expressed in terms of a constant depending on the probabilities of activating each agent and such that $Q \leq 1.6(\alpha_G + 1)$. The result holds for any differentiable function $g\colon \mathcal{X} \to \mathbb{R}$, σ-strongly convex with respect to some norm $\|\cdot\|$.

Theorem 2.3. *Consider a network $G = (V, E)$ of N agents. Assume that, at each time step t each agent v is independently activated with probability $q_v \in [0,1]$. If all agents run OMD with an oblivious network interface and using $g_t = \frac{\sqrt{t}}{\eta} g$, for $\eta > 0$, the network regret satisfies*

$$\mathbb{E}[R_T] \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{QT}$$

where $Q = \sum_{v\in V'}(q_v c_v)/Q_v$, $D \geq \sup g$, $L \geq \sup \|\nabla \ell_t\|_*$, and $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. In particular, choosing $\eta = \sqrt{2\sigma D}/L$ gives $\mathbb{E}[R_T] \leq L\sqrt{2DQT/\sigma}$.

Proof. Fixing an arbitrary $\boldsymbol{x} \in \mathcal{X}$, setting $r_t(v) = \ell_t\big(\boldsymbol{x}_t(v)\big) - \ell_t(\boldsymbol{x})$, and proceeding as in Theorem 2.2 yields, for each $v \in V'$,

$$\mathbb{E}\left[\sum_{t=1}^T r_t(v)\right] \leq \left(\frac{D}{\eta} + \frac{\eta L^2}{2\sigma}\right)\sqrt{\frac{T}{Q_v}} \tag{2.6}$$

Now we write $\mathbb{E}[R_T] = \sup_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\big[R_T(\boldsymbol{x})\big]$, where

$$\begin{aligned}
\mathbb{E}\big[R_T(\boldsymbol{x})\big] &= \mathbb{E}\left[\sum_{t=1}^T \frac{1}{\sum_{w\in V} X_t(w)} \sum_{v\in V'} r_t(v) X_t(v)\right] \\
&= \sum_{t=1}^T \sum_{v\in V'} \mathbb{E}\left[\frac{X_t(v)}{\sum_{w\in V} X_t(w)}\right] \mathbb{E}\big[r_t(v)\big] \\
&= \sum_{v\in V'} q_v c_v \sum_{t=1}^T \mathbb{E}\big[r_t(v)\big] \qquad (2.7)
\end{aligned}$$

and the last identity follows by Lemma 2.2. Putting identity (2.7) and inequality (2.6) together gives

$$\mathbb{E}\big[R_T(\boldsymbol{x})\big] \le \left(\sum_{v\in V'} q_v c_v \sqrt{\frac{1}{Q_v}}\right)\left(\frac{D}{\eta}+\frac{\eta L^2}{2\sigma}\right)\sqrt{T} \le \sqrt{\sum_{v\in V'}\frac{q_v c_v}{Q_v}}\left(\frac{D}{\eta}+\frac{\eta L^2}{2\sigma}\right)\sqrt{T}$$

where in the last inequality we used Jensen inequality and $\sum_{v\in V'} q_v c_v \le 1$. This concludes the proof. $\square$

We now prove that the inequality $Q(q) \le \alpha_G$ is always true up to a small constant factor.

Lemma 2.3. *Let $G = (V, E)$ be an undirected graph. For all $v \in V$, choose numbers $q_v \in (0, 1]$ and define c_v and Q_v as in* (2.4) *and* (2.5) *respectively. Then*

$$Q = \sum_{v\in V}\frac{q_v c_v}{Q_v} \le \frac{\alpha_G + 1}{1-e^{-1}}$$

Proof. Let $P_v = \sum_{w\in\mathcal{N}_v} q_w$, $V_1 = \big\{v \in V \mid P_v \ge 1\big\}$, and $V_0 = \big\{v \in V \mid P_v < 1\big\}$. We begin by splitting the sum as follows

$$\sum_{v\in V}\frac{q_v c_v}{Q_v} = \sum_{v\in V_1}\frac{q_v c_v}{Q_v} + \sum_{v\in V_0}\frac{q_v c_v}{Q_v}$$

We upper bound the two terms separately. Since the minimum $\min_{v\in V_1} Q_v$ is attained when $q_v = 1/|\mathcal{N}_v|$ for all $v \in \mathcal{N}_v$, we can lower bound, for each $v \in V_1$,

$$Q_v \ge 1 - \left(1 - \frac{1}{|\mathcal{N}_v|}\right)^{|\mathcal{N}_v|} \ge 1 - e^{-1}$$

This together with $\sum_{v\in V} q_v c_v \le 1$ yields

$$\sum_{v\in V_1}\frac{q_v c_v}{Q_v} \le \frac{1}{1-e^{-1}}$$

To upper bound the sum over V_0, we first use the inequality $1 - x \le e^{-x}$ that holds for all $x \in [0, 1]$. Setting $x = q_w$ gives

$$Q_v \ge 1 - \exp\left(-\sum_{w\in\mathcal{N}_v} q_w\right) = 1 - e^{-P_v}$$

For all $v \in V_0$, we can then use the inequality $1 - e^{-x} \ge (1 - e^{-1})x$, holding for all $x \in [0, 1]$. Setting $x = P_v < 1$ we conclude that $Q_v \ge (1 - e^{-1})P_v$ for all $v \in V_0$. Finally, using $c_v \le 1$ we can write

$$\sum_{v\in V_0}\frac{c_v q_v}{Q_v} \le \frac{1}{1-e^{-1}}\sum_{v\in V}\frac{q_v}{P_v} \le \frac{\alpha_G}{1-e^{-1}}$$

where the last inequality follows by Lemma 2.2. Putting everything together gives the result. $\square$

Chapter 3

Online non-Concave Maximization

In this chapter we propose a new analysis of the Piyavskii–Shubert algorithm that holds for non-concave maximization over non-compact d-dimensional domains. More precisely, we address the problem of finding a global maximizer of a function $f : D \to \mathbb{R}$ satisfying a very mild regularity condition, using as few (possibly noisy) evaluations of f as possible. We first derive new regret bounds for the classical Piyavskii–Shubert algorithm when evaluations are deterministically perturbed. They match the best known upper bounds in terms of the so called *near-optimality dimension* of f. We then use these results to design variants of the algorithm that, for any given ε, stop automatically returning an ε-optimal point both under deterministic and stochastic perturbations.

3.1 Introduction

The goal of online optimization is to find an approximate maximizer of a given function $f\colon D \subset \mathbb{R}^d \to \mathbb{R}$ with as little evaluations of f as possible. In this chapter we assume that the only access to the function f is through an oracle returning the (possibly) perturbed values of the function at the queried points. Perturbations can be deterministic or stochastic. No analytical expression of f or any of its derivatives is available.

At each round k the learner picks a new point $x_k \in D$ and the value $f(x_k)$ is revealed by the oracle, up to an additive perturbation ξ_k. After each evaluation, the learner can return a point $x_k^\star \in D$, which may differ from the last queried point x_k.

We measure the accuracy of the approximation provided by the point $x_n^\star$ returned after the n-th evaluation of the function with the relative error

$$r_n := \sup(f) - f(x_n^\star) \tag{3.1}$$

Following the bandit optimization literature, we call this error *simple regret* (or *regret* for short).

We consider two variants of the problem. In the first one, a budget n of evaluations is given to the learner. The goal is to find an $x_n^\star \in D$ such that $r_n \leqslant \varepsilon$, with $\varepsilon > 0$ as small as possible. In the second one, a level of accuracy $\varepsilon > 0$ is given instead. In this case, the goal is to find an $x_n^\star \in D$ such that $r_n \leqslant \varepsilon$, with n as small as possible.

Due to numerous practical applications, this black-box global optimization problem has received considerable attention over the past decades. Many different algorithms have been proposed in several communities such as concave optimization [69, 14, 17], non-concave optimization [41, 49, 46], stochastic optimization or approximation [88, 82], Bayesian optimization [15], and bandit optimization over metric spaces [67].

In this article, we focus on the case where f attains its maximum at some $x^\star \in D$ and it is Lipschitz around $x^\star$ (Assumption 3.1). Unlike global Lipschitzness, this assumption do not imply continuity anywhere but at the maximizer $x^\star$ (Figure 3.1). We consider two different settings. In the deterministic setting the values of f are observed up to deterministic (and possibly adaptive) adversarial perturbations (Section 3.3). In the stochastic setting the queried values are observed up to a subgaussian noise (Section 3.4). We extend a classical algorithm designed by Piyavskii [73] and Shubert [86] for the deterministic setting with no perturbations. We call it *the Piyavskii–Shubert algorithm* in the sequel. The principle is simple: at each round k, the query point x_k is chosen as a maximizer of a proxy function $\widehat{f}_k \colon D \to \mathbb{R}$ that provably satisfies $\widehat{f}_k(x^\star) \geqslant f(x^\star)$. After observing the perturbed value of $f(x_k)$, the learner updates the proxy function $\widehat{f}_{k+1}$ and uses it to choose the next point x_{k+1}.

Several papers studied the Piyavskii–Shubert algorithm in the eighties and nineties (see, e.g., [60, 64] or the survey by Hansen et al. [41]). Despite this literature, little was known about the rate of convergence of the simple regret r_n as a function of the number n of evaluations of the function.

Most papers establish a convergence guarantee (with no rate) under a global Lipschitz assumption on the function f. A notable exception is in dimension $d = 1$, where a sharp bound on the sample complexity was derived by Hansen et al. [40] for a variant of the Piyavskii–Shubert algorithm stopped as a function of a target precision $\varepsilon > 0$. More precisely, they proved that the number of iterations required by the algorithm to reach precision ε is at most proportional to $\int_0^1 \bigl(f(x^\star) - f(x) + \varepsilon\bigr)^{-1} \mathrm{d}x$, for some tight multiplicative constant. In this chapter these authors rely heavily on the one-dimensional setting to study the proxy functions $\widehat{f}_k$ in an explicit manner. In the same paper, they claim that *"Extending the results of [their] paper to the multivariate appears to be difficult"*.

3.1.1 Main contributions

In this chapter we provide a theoretical analysis of the simple regret r_n of the Piyavskii–Shubert algorithm in arbitrary dimension $d \geqslant 1$, using recent concepts from the bandit optimization literature. Our only assumptions are that f is Lipschitz around a maximizer $x^\star$ (see Section 3.2 for a rigorous definition) and its domain D is bounded.

Deterministic setting. For the deterministic setting, we prove the first non-trivial[1] upper bound on the simple regret r_n of the Piyavskii–Shubert algorithm in arbitrary dimension $d \geqslant 1$. This bound involves a quantity $d^\star \in [0, d]$ usually called the *near-optimality dimension* of f (see Section 3.5.1 for a definition) and when the values of f can be observed without perturbations is roughly of the form

$$r_n \lesssim \begin{cases} n^{-1/d^\star} & \text{if } d^\star > 0 \\ \exp\bigl(-\Omega(n)\bigr) & \text{if } d^\star = 0 \end{cases} \tag{3.2}$$

This regret bound matches, up to logarithmic factors, the best bound known so far[2] in this deterministic setting. Examples of other algorithms that attain this bound include the branch-and-bound algorithm of [72], the DOO algorithm of [66], and the LIPO algorithm by [56] (see Section 3.1.2 below for further details).

To prove (3.2), we upper bound the sample complexity of the algorithm. Our sample complexity bounds holds even if the values of f are perturbed, provided that the absolute value of the perturbations is bounded by a known constant $\alpha \geqslant 0$. More precisely, we derive the following equivalent upper bound: if $\alpha = \mathcal{O}(\varepsilon)$, the number n of iterations needed to reach precision ε is at most of $n = \mathcal{O}\bigl((1/\varepsilon)^{d^\star}\bigr)$ for $d^\star > 0$ or $n = \mathcal{O}\bigl(\log(1/\varepsilon)\bigr)$ for $d^\star = 0$.

This is satisfactory if an overall budget n is imposed, but it might be impractical if a target precision ε is required instead. In this case, the knowledge of $d^\star$ is necessary to stop the algorithm at the right time, i.e., to compute the upper bound $n = \mathcal{O}\bigl((1/\varepsilon)^{d^\star}\bigr)$ or $n = \mathcal{O}\bigl(\log(1/\varepsilon)\bigr)$ on the number of iterations needed to reach precision ε.

[1] A crude regret bound of the form $r_n = \mathcal{O}\bigl(n^{-1/d}\bigr)$, where d is the ambient dimension, can be obtained from [64, Theorem 4.2] when f is globally Lipschitz. The authors show that the Piyavskii–Shubert algorithm is minimax optimal among all algorithms, and therefore superior to a uniform grid search ensuring a simple regret of order $n^{-1/d}$.

[2] We are not aware of lower bounds for all possible values of $d^\star \in [0, d]$ except for $d^\star = d$ [69, Theorem 1.1.2] or for $d^\star = d/2$ [44, Theorem 9]. However we conjecture that the best sample complexity upper bounds known so far are minimax optimal (up to logarithmic factors).

To address this issue, we study a version of the Piyavskii–Shubert algorithm that takes ε as input and stops automatically after a simple condition is satisfied (Algorithm 11), as in the one-dimensional study by [40]. We prove that this algorithm is adaptive to the near-optimality dimension $d^\star$: it stops after $n = \mathcal{O}\big((1/\varepsilon)^{d^\star}\big)$ or $n = \mathcal{O}\big(\log(1/\varepsilon)\big)$ iterations (if $d^\star > 0$ or $d^\star = 0$ respectively) without any prior knowledge on $d^\star$, and guarantees $r_n \leqslant \varepsilon$.

Stochastic setting. For the stochastic setting, we design a natural extension of the Piyavskii–Shubert algorithm based on mini-batch sampling, a technique in which each value of f is queried a small number of times in a row in order to compute a reasonable estimate of its mean. We study the number n of evaluations of f needed by the algorithm to reach precision ε with probability at least $1-\delta$. Note that the number of evaluations of f is now larger than the number of iterations of the algorithm because of the mini-batch sampling. We derive sample complexity bounds of the form $n = \widetilde{\mathcal{O}}\big((1/\varepsilon)^{d^\star+2}\log(1/\delta)\big)$, where the $\widetilde{\mathcal{O}}$ notation hides logarithmic factors in $1/\varepsilon$. These bounds correspond to regret upper bounds roughly of the form

$$r_n \lesssim n^{-1/(d^\star+2)}$$

which are known to be worst-case optimal as a consequence of the minimax lower bound of Bubeck et al. [19, Theorem 13] (this lower bound is stated for the cumulative regret, but it can be adapted for the simple regret).

Examples of other algorithms with a similar regret bound in the stochastic setting include, e.g., the zooming algorithm by [53], the HOO algorithm by [19] and the StoOO algorithm by [67] (some of these papers derived upper bounds on the cumulative regret, which in turn imply upper bounds on the simple regret).

3.1.2 Earlier works: some further comments

Over the last decade the Lipschitz global optimization problem has received a lot of attention in the bandit optimization community. We list below some of the algorithms that were inspired from the Piyavskii–Shubert algorithm but were analyzed using more recent techniques. We borrowed several concepts from this literature (as, e.g., the near-optimality dimension $d^\star$) to get back to the original Piyavskii algorithm.

DOO A simple algorithm in the deterministic setting is the algorithm DOO (Deterministic Optimistic Optimization) [66]. It is very similar to the Piyavskii–Shubert algorithm, except that the proxy functions $\widehat{f}_t$ are piecewise-constant instead of piecewise-conic, with pieces that correspond to cells of a predefined hierarchical partition of D. These piecewise-constant proxy functions are slightly weaker in terms of approximation of f, but offer a huge advantage in terms of computational complexity. Indeed, using a max-heap structure to sequentially compute a maximizer of $\widehat{f}_k$, the DOO algorithm can be implemented in $\mathcal{O}(n\log n)$ total running time after n iterations. In contrast, it is conjectured that an exact computation of $\arg\max \widehat{f}_k$ for the Piyavskii–Shubert algorithm might require a running time exponential in the dimension d (see [64] or [60] for a partial analysis of the associated exact optimization problem).

Despite this conjectured computational issue in high dimensions d, the Piyavskii–Shubert algorithm is an important algorithm that is both very natural and optimal in the way the proxy functions $\widehat{f}_k$ approximate f (see [41, Theorem 4]). We believe that the tighter piecewise-conic proxy functions that it uses (in contrast to the looser piecewise-constant proxy functions of DOO) could reduce the sample complexity by a multiplicative constant larger than 1. When the near-optimality dimension vanishes, this would lead to a regret bound $r_n \lesssim \exp(-\Omega(n))$ with this constant *inside* the exponential. We leave this challenging question for future works.

SOO The previous algorithm has been extended [66, Simultaneous Optimistic Optimization (SOO)] by not requiring the knowledge of the semi-metric $\ell(x,y)$ defining the regularity of f while still achieveing almost the same theoretical performance as DOO when $d^\star \neq 0$. At first, it might seem to be a huge advantage over algorithms like the Piyavskii–Shubert algorithm that strongly rely on the knowledge of this metric. However, as noticed in [39], the SOO algorithm is only partially adaptive. Indeed the success of approaches based on hierarchical partitioning relies on a central assumption: the partitioning must be *well-shaped* with respect to the regularity of f. This implies that ℓ should be known at least approximately. In this sense, then, the hierarchical partitioning already contains such regularity information.

POO Grill et al. [39] used this observation to build a truly adaptive algorithm (POO). Noting that earlier papers only use the semi-metric ℓ to relate the function f and the hierarchical partitioning $\mathcal{P}$ at hand, they replace two assumptions (one relating f to ℓ, and another relating ℓ to $\mathcal{P}$) with a single assumption connecting f and $\mathcal{P}$. They then show that for any function satisfying a local smoothness assumption with respect to $\mathcal{P}$, the POO algorithm solves the optimization problem almost as quickly as if the smoothness of f was known in advance. In this chapter we decide to make full use of the metric information. The Piyavskii–Shubert algorithm requires the knowledge of the semi-metric (a norm, in our case). As shown in [41], this enables the use of a sharp proxy function to drive the optimization process. They provide a set of numerical experiment in dimension one suggesting a clear improvement of the empirical performance induced by of such a modification of the procedure with respect to DOO-like algorithms. Our goals are both to search for a theoretical foundation of this claim and to determine extensions of this approach that hold in the stochastic setting.

Moreover, we provide two natural extensions of our algorithms (Algorithms 11 and 12) that do not require a prior knowledge of the near-optimality dimension $d^\star$ in order to guarantee an ε-optimal solution (the number n of evaluations after which these algorithms stop is chosen automatically as a function of the observations). We conjecture that such extensions could also be provided for algorithms of the OO family.

3.1.3 Outline of the chapter

This chapter is organized as follows. We state our main assumptions on f and prove a few useful lemmas in Section 3.2. In Section 3.3 we revisit the Piyavskii–Shubert algorithm in the deterministic setting with perturbations by proving state-of-the-art regret bounds in terms of the intrinsic dimension $d^\star$ of f, for an arbitrary dimension $d \geqslant 1$. We study two types of algorithms: the natural extension of the original Piyavskii–Shubert algorithm, whose optimal stopping would require the knowledge of $d^\star$, and an adaptive variant that stops automatically guaranteeing precision ε when a halting criterion is met. In Section 3.4, we design an extension of the Piyavskii–Shubert algorithm for the stochastic setting and derive minimax optimal high-probability bounds on the simple regret. Finally, in Section 3.5.4 we discuss the connections with a regret bound of Hansen et al. [40] in dimension $d = 1$.

3.2 Assumption, definitions, and notation

In this section we introduce a regularity assumption, all notation, and all definitions that will be used throughout the chapter.

3.2.1 Assumption on f

In all the sequel, D is any nonempty bounded (not necessarily compact) subset of $\mathbb{R}^d$, $f\colon D \to \mathbb{R}$ is any function, and $\|\cdot\|$ is any norm on $\mathbb{R}^d$. We make the following assumption, which is sometimes referred to as "local smoothness" [67].

Assumption 3.1 (Lipschitzness around a maximum). *We assume that f attains its maximum $f^\star$ at some $x^\star \in D$ and that there exists a constant $L_0 > 0$ such that, for all $x \in D$,*

$$f(x) \geqslant f^\star - L_0 \|x^\star - x\| .$$

Moreover, we assume that only D, $\|\cdot\|$, and an upper bound L_1 on L_0 are known to the learner.

Note that we do not even require this assumption to be true for *all* maximizers. To the best of our knowledge, all previous work on the Piyavskii–Shubert algorithm and variants thereof assume f to satisfy

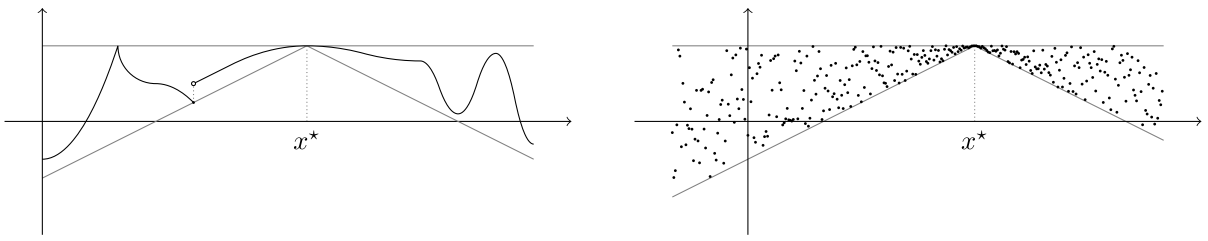

Figure 3.1: On the left, a typical example of a function Lipschitz around a maximum $x^\star$: it has multiple maxima, points with vertical tangents, and discontinuities. On the right, a pathological example of function discontinuous everywhere but at a maximum $x^\star$, but still Lipschitz around $x^\star$. Our analysis holds even in this extreme case.

some *global* continuity condition (e.g., Lipschitzness, Hölderness, uniform continuity) [92, 73, 40, 74, 54, 33, 85, 68]. However, recent contributions from the bandit optimization literature [67] have shown that the behavior of this type of algorithms is usually driven by the regularity of f around its maxima. Our analysis will show that this is also the case for the Piyavskii–Shubert algorithm.

Lipschitzness around a maximum is a significantly weaker assumption than global Lipschitzness. The only constraint that it poses to the function f is for its graph to lie between the cone $\big\{(x,y) \in \mathbb{R}^{d+1} : y = f(x^\star) - L_0\|x^\star - x\|\big\}$ and the hyperplane $\big\{(x,y) \in \mathbb{R}^{d+1} : y = f(x^\star)\big\}$. Most of the good properties that globally Lipschitz functions enjoy are not guaranteed under this assumption. For example, f could be non-differentiable or even discontinuous *everywhere* on $D \setminus \{x^\star\}$ (Figure 3.1).

Furthermore, our analysis will prove that the boundedness of D is not only necessary for the Piyavskii–Shubert algorithm to be well-defined, but it is also sufficient to guarantee its convergence to an optimum. This shows in particular that the common requirement that D is compact [73, 40, 74, 54, 33, 85, 68] can be weakened by dropping the closure assumption.

3.2.2 Useful notation and definitions

We denote the set of integers by $\mathbb{Z}$, the set of nonnegative integers $\{0, 1, 2, \dots\}$ by $\mathbb{N}$, and the set of positive natural numbers $\{1, 2, \dots\}$ by $\mathbb{N}^*$. For all $x \in \mathbb{R}$, we write $\lceil x \rceil$ for the value $\min\{k \in \mathbb{Z} : k \geq x\}$ of the ceiling function at x. For all $\delta > 0$, we denote by $B_{\|\cdot\|}(\delta)$ the ball of radius δ in $\big(\mathbb{R}^d, \|\cdot\|\big)$ centered at the origin:

$$B_{\|\cdot\|}(\delta) := \big\{x \in \mathbb{R}^d : \|x\| \leqslant \delta\big\} .$$

We now recall the definitions of packing and covering numbers. The former is of utmost importance to our analysis. For any bounded set $A \subset \mathbb{R}^d$ and any real number $r > 0$:

- the *r-packing number* of A is the largest number of r-separated points contained in A, that is,

$$\mathcal{N}(A, r) := \sup\left\{k \in \mathbb{N}^* : \exists x_1, \dots, x_k \in A, \min_{i \neq j} \|x_i - x_j\| > r\right\} , \tag{3.3}$$

  with the convention that $\mathcal{N}(\varnothing, r) := 0$;

- the *r-covering number* of A is the smallest cardinality of an r-covering of A, that is,

$$\mathcal{M}(A, r) := \min\big\{N \in \mathbb{N}^* : \exists x_1, \dots, x_N \in \mathbb{R}^d, \forall x \in A, \exists i \in \{1, \dots, N\}, \|x - x_i\| \leqslant r\big\} ,$$

  with the convention that $\mathcal{M}(\varnothing, r) := 0$.

In Section 3.5.1 we recall a few known inequalities about packing and covering numbers that will prove useful throughout the whole chapter.

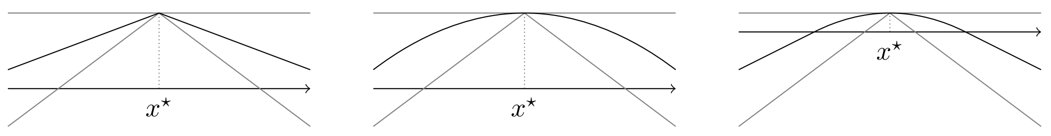

Figure 3.2: On the left, a linear function ($d^\star = 0$). In the center, a quadratic function ($d^\star = d/2$). On the right, a function that linear outside of a neighborhood of $x^\star$ ($d^\star = 0$ for large r), but quadratic inside ($d^\star = d/2$ for small r).

3.2.3 Sets of near-optimal points, with examples

In this section we will introduce a few important definitions and properties that will play a crucial role in our analysis. For all $\varepsilon > 0$, we define the set of *ε-optimal points* of $f\colon D \to \mathbb{R}$ by

$$\mathcal{X}_\varepsilon := \left\{ x \in D : f(x) \geqslant f(x^\star) - \varepsilon \right\} .$$

We also denote its complement (i.e., the set of *ε-suboptimal points*) by $\mathcal{X}_\varepsilon^c$ and, for all $0 \leqslant a < b$, we define the $(a,b]$-*layer*

$$\mathcal{X}_{(a,b]} := \mathcal{X}_a^c \cap \mathcal{X}_b = \left\{ x \in D : a < f(x^\star) - f(x) \leqslant b \right\} . \tag{3.4}$$

Whenever the explicit dependence on ε or $(a,b]$ can be omitted, we will simply refer to these sets as sets of *near-optimal points*, sets of *suboptimal points*, or *layers*. In particular, we will say that an $(a,b]$-layer is a *suboptimal layer* if $a > 0$.

Since f is L_0-Lipschitz around $x^\star$, every point in D is ε_0-optimal with ε_0 defined by

$$\varepsilon_0 := L_0 \sup_{x,y \in D} \|x - y\| . \tag{3.5}$$

In other words, $\mathcal{X}_{\varepsilon_0} = D$. For this reason, without loss of generality we will only consider values of ε smaller than ε_0.

As we will see in Sections 3.3 and 3.4, the size of the sets of near-optimal points and that of layers will be key quantities in our sample complexity bounds. As it turns out, the "correct" notion of size for this problem is the packing number (3.3). In particular, we will derive explicit and immediate corollaries whenever, for some $C^\star, d^\star \geqslant 0$, we have:[3]

$$\forall r \in (0, \varepsilon_0), \qquad \mathcal{N}\left(\mathcal{X}_r, \frac{r}{2L_0} \right) \leqslant C^\star \left(\frac{\varepsilon_0}{r} \right)^{d^\star} . \tag{3.6}$$

The idea behind Inequality (3.6) is that ε-optimal points are hard to find if the corresponding set $\mathcal{X}_\varepsilon^c$ of ε-suboptimal points is large. Since for any increasing sequence $\varepsilon := r_0 < r_1 < r_2 < \dots$, the set of ε-suboptimal points $\mathcal{X}_\varepsilon^c$ can be decomposed into a union of suboptimal layers $\mathcal{X}_{(r_0,r_1]}, \mathcal{X}_{(r_1,r_2]}, \mathcal{X}_{(r_2,r_3]}, \dots$, and each of these layers $\mathcal{X}_{(r_{s-1},r_s]}$ is included in $\mathcal{X}_{r_s}$ (by definition of layer (3.4)), by controlling the size of each of these $\mathcal{X}_{r_s}$ we can control the size of $\mathcal{X}_\varepsilon^c$. Therefore, by controlling how large the sets $\mathcal{X}_r$ can be at all scales r, the parameters $C^\star$ and $d^\star$ quantify the difficulty of the optimization problem. (See the discussion in Section 3.5.2 about the related notions of *near-optimality dimension* [19] and *zooming dimension* [53].) As noted in Lemma 3.7 (Section 3.5.2), Inequality (3.6) is always true with $C^\star = 9^d$ and $d^\star = d$. However, depending on f, significantly smaller values of the constants $C^\star$ and $d^\star$ may be picked. We provide three examples below. The last one of them indicates that a "worst-case" single value of $d^\star$ may be insufficient to give an appropriate description of the hardness of the optimization problem.

[3]Property (3.6) could be rewritten equivalently as $\forall r \in (0, \varepsilon_0)$, $\mathcal{N}(\mathcal{X}_r, r) \leqslant C^\star \left(\frac{1}{r}\right)^{d^\star}$. Our choice of normalizing r by $2L_0$ (resp., by ε_0) makes $C^\star$ more "intrinsic" (typically independent of L_0), as can be seen from the examples below.

Example 3.1 (Linear Regime)**.** *Consider any norm $\|\cdot\|$ and the function $f(x) = 1 - L_0\|x-a\|$ on any bounded domain $D \subset \mathbb{R}^d$, with $a \in D$ (Figure 3.2, left). Then f is L_0-Lipschitz and, for all $r \in (0, \varepsilon_0)$, we have $\mathcal{X}_r = \big\{x \in D : \|x-a\| \leqslant r/L_0\big\}$, which gives*

$$\begin{aligned}
\mathcal{N}\left(\mathcal{X}_r, \frac{r}{2L_0}\right) &\leqslant \mathcal{N}\left(B_{\|\cdot\|}(r/L_0), \frac{r}{2L_0}\right) \\
&\leqslant \mathcal{M}\left(B_{\|\cdot\|}(r/L_0), \frac{r}{4L_0}\right) && \text{(by (3.17))} \\
&\leqslant \left(1 + \frac{2(r/L_0)}{r/(4L_0)}\right)^d && \text{(by (3.18))} \\
&= 9^d \,,
\end{aligned}$$

which does not depend on r. Thus Inequality (3.6) holds with $C^\star = 9^d$ and $d^\star = 0$.

Example 3.2 (Quadratic Regime)**.** *Fix any $\beta > 0$. Consider the Euclidean norm $\|\cdot\|_2$ on $\mathbb{R}^d$ and the function $f(x) = 1 - \beta\|x-a\|_2^2$ on any bounded domain $D \subset \mathbb{R}^d$, with $a \in D$ (Figure 3.2, center). Then, for all $r \in (0, \varepsilon_0)$, $\mathcal{X}_r = \big\{x \in D : \|x-a\|_2 \leqslant \sqrt{r/\beta}\big\}$. Letting $\alpha = \sup_{x\in D}\|x-a\|_2$ and $L_0 = 2\alpha\beta$, note that f is L_0-Lipschitz with respect to $\|\cdot\|_2$. Then, for all $r \in (0, \varepsilon_0)$,*

$$\begin{aligned}
\mathcal{N}\left(\mathcal{X}_r, \frac{r}{2L_0}\right) &\leqslant \mathcal{N}\left(B_{\|\cdot\|_2}\big(\sqrt{r/\beta}\big), \frac{r}{4\alpha\beta}\right) \\
&\leqslant \mathcal{M}\left(B_{\|\cdot\|_2}\big(\sqrt{r/\beta}\big), \frac{r}{8\alpha\beta}\right) && \text{(by (3.17))} \\
&\leqslant \left(1 + \frac{2\sqrt{r/\beta}}{r/(8\alpha\beta)}\right)^d = \left(1 + 16\alpha\sqrt{\frac{\beta}{r}}\right)^d && \text{(by (3.18))} \\
&\leqslant \left(1 + 8\sqrt{2}\right)^d \left(\frac{\varepsilon_0}{r}\right)^{d/2} . && \text{(since } \varepsilon_0 = 2\alpha\beta\sup_{x,y\in D}\|x-y\|_2\text{)}
\end{aligned}$$

Thus Inequality (3.6) holds with $C^\star = \big(1 + 8\sqrt{2}\big)^d$ and $d^\star = d/2$.

The next example shows that a unique value of $d^\star$ is sometimes insufficient to describe the shape of a function around a maximizer. Notably, our regret bounds in the next sections will not depend on a single (worst-case) value of $d^\star$, but on a suitable combination of such values at different scales r. This allows to give tighter bounds on the regret in cases like the following one.

Example 3.3 (Mixed Regime)**.** *Consider the euclidean norm $\|\cdot\|_2$ on $\mathbb{R}^d$ and the function $f\colon [-1,1]^d \to \mathbb{R}$ defined by*

$$f(x) = \begin{cases} 1/4 - \|x\|_2^2 & \textit{if } \|x\|_2 \leqslant 1/2 \\ 1/2 - \|x\|_2 & \textit{if } \|x\|_2 > 1/2 \end{cases}$$

(Figure 3.2, right). A direct verification shows that f is 1-Lipschitz and attains its maximum at $x^\star = 0$, with $f(x^\star) = 1/4$. In this case, $\varepsilon_0 = \sup_{x,y}\|x-y\|_2 = 2\sqrt{d}$. Proceeding as in the previous two examples, we get, for all $r \in \big[1/4, 2\sqrt{d}\big)$,

$$\mathcal{N}\left(\mathcal{X}_r, \frac{r}{2L_0}\right) \leqslant 2 \cdot 17^d \,,$$

but for all $r \in (0, 1/4)$,

$$\mathcal{N}\left(\mathcal{X}_r, \frac{r}{2L_0}\right) \leqslant \left(\frac{1 + 4\sqrt{2}}{\sqrt{d}}\right)^d \left(\frac{\varepsilon_0}{r}\right)^{d/2} .$$

Therefore, Inequality (3.6) holds with $d^\star = 0$ for large values of r (linear regime) or with $d^\star = d/2$ for small values of r (quadratic regime). Many different examples can be designed this way.

3.3 Deterministic perturbations

In this section we provide a new regret analysis of the Piyavskii–Shubert algorithm [73, 86] for the *deterministic setting*, i.e., when values of f are observed up to deterministic perturbations with absolute value bounded by a known constant. These perturbations can be chosen arbitrarily and even adaptively to the learner's algorithm. In Section 3.4 we will study the *stochastic setting*, in which values of f are observed up to a subgaussian noise.

We consider two variants of the problem in the deterministic setting. When an overall budget n on the number of evaluations of f is fixed in advance, we present and analyze Algorithm 10 (Section 3.3.1), which makes n queries to f before returning a near-optimal point. When an accuracy level ε is imposed instead, we introduce and study Algorithm 11 (Section 3.3.2), which stops automatically and guarantees an ε-optimal solution after stopping. In the first case, we upper bound the number n of iterations required to reach any given precision $\varepsilon > 0$ (Theorem 3.1 and Corollary 3.1). This leads to an upper bound on the regret as a function of the number n of queries to f (Corollary 3.2). In the second case, we upper bound the number n' of evaluations of f after which the algorithm automatically stops (Theorem 3.2 and Corollary 3.3). The second case is more powerful, since n' is of the same order of magnitude as n but the knowledge of n is not required to guarantee an ε-optimal solution.

3.3.1 Budget on queries

In this section we derive a new bound for the simple regret of the Piyavskii–Shubert algorithm as a function of the number n of queries to f (Corollary 3.2). Formally, the algorithm applies to the following online optimization protocol.

A function $f\colon D \subset \mathbb{R}^d \to \mathbb{R}$ that attains its maximum at some point $x^\star \in D$ is fixed in advance. Only the domain D of the function, an upper bound $\alpha \geqslant 0$ on the absolute value of the perturbations (see below), and the total number of queries $n \in \mathbb{N}^\star$ are known to the learner in advance.

For each $k = 1, \dots, n$:

1. the learner chooses a point $x_k \in D$,
2. the environment picks a deterministic perturbation
$$\xi_k \in [-\alpha, \alpha] \,, \tag{3.7}$$
depending on f and all points x_s chosen by the learner up to and including the current one x_k,
3. the value $f(x_k) + \xi_k$ is revealed to the learner.

The learner then outputs a prediction $x_n^\star$ with the goal of minimizing the simple regret
$$r_n := f(x^\star) - f(x_n^\star) \,. \tag{3.8}$$

In this section we extend the simpler original version of the Piyavskii–Shubert algorithm designed for a fixed query budget n to our non-compact d-dimensional setting with perturbations. The behavior of the algorithm is illustrated in Figure 3.3. The Piyavskii–Shubert algorithm (Algorithm 10) works by maintaining a proxy $\widehat{f}_k$ of the objective function f that (up to perturbations) upper bounds f at least at the maximizer $x^\star$. After each new evaluation, the graph of the proxy gets closer and closer to that of the objective. The points that are queried by the algorithm are those where the $\widehat{f}_k$ is biggest. The intuition is that if $\widehat{f}_k$ is sufficiently close to f, then the point with the highest value among all proxy functions should also be near-optimal for f.

Note that in Algorithm 10, we pick x_{k+1} as an α-optimal point of $\widehat{f}_k$. The algorithm could be defined slightly more generally by picking x_{k+1} as an η-optimal point of $\widehat{f}_k$, where η is an additional parameter independent of the bound α on $|\xi_k|$. Our choice to lighten the notation is due to the fact that if $\alpha > 0$, in order to have a meaningful result, η should eventually be set to $\mathcal{O}(\alpha)$ anyway. Moreover, with our choice,

Algorithm 10: Piyavskii–Shubert algorithm (with known query budget n)

```
Input: Lipschitz constant $L_1$, number of iterations $n$, perturbation scale $\alpha \geqslant 0$, initial guess $x_1 \in D$.
1 for $k = 1, \dots, n$ do
2     observe $y_k = f(x_k) + \xi_k$;
3     update $\widehat{f}_k(x) = \min_{i \in \{1,\dots,k\}} \{ y_i + L_1 \|x_i - x\| + \alpha \}$;
4     pick $x_{k+1} \in D$ such that $\widehat{f}_k(x_{k+1}) \geqslant \sup_{x \in D} \widehat{f}_k(x) - \alpha$;
5 compute $i_n^\star = \arg\max_{i \in \{1,\dots,n\}} y_i$;
6 return $x_n^\star := x_{i_n^\star}$;
```

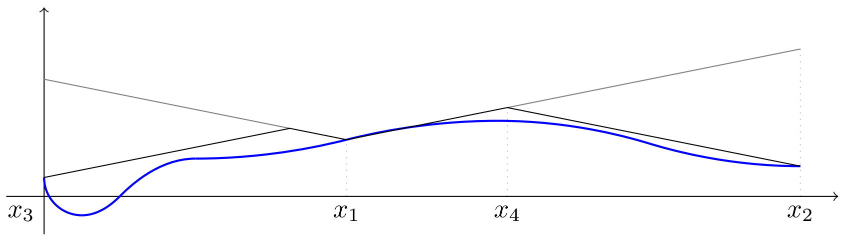

Figure 3.3: First three iterations of the Piyavskii–Shubert algorithm in dimension $d = 1$. In blue the function f. In black the proxy function $\widehat{f}_3$.

the case $\alpha = 0$ still includes the classic setting in which D is compact and values are observed with no perturbations.

The main results of this section are Theorem 3.1 and its Corollaries 3.1 and 3.2. We will address the version with an automatic stopping criterion in Section 3.3.2.

We now upper bound the minimum number of iterations (i.e., the sample complexity) needed for the simple regret $r_n = f(x^\star) - f(x_n^\star)$ to fall below some threshold ε. The following theorem is proved in Section 6.

Theorem 3.1. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $n \in \mathbb{N}^\star$, $\alpha \in [0, \varepsilon/24)$, $x_1 \in D$, and assume that the Piyavskii–Shubert algorithm (Algorithm 10) is run with inputs L_1, n, α, x_1. Defining*

$$\widetilde{n} := \sum_{s=0}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil - 1} \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-s-1}, \varepsilon_0 2^{-s}]}, \frac{\varepsilon_0 2^{-s-1} - 3\alpha}{L_1} \right) + 1 \,, \tag{3.9}$$

if $n \geqslant \widetilde{n}$, then the simple regret (3.1) of the Piyavskii–Shubert algorithm satisfies $r_n \leqslant \varepsilon + 2\alpha$.

Note that the previous bound, expressed in terms of the packing numbers of increasingly better and better layers, is tight. If the objective function is constant all these packing numbers vanish and we are left with $\widetilde{n} = 1$, which is indeed the minimum number of evaluations needed to return an ε-maximizer (because all points are maximizers). This result is therefore stronger than usual bounds expressed in terms of the near-optimality dimension. Nevertheless, bounds in terms of the near-optimality dimensions are more readable. Therefore we now upper bound the previous result as a function of the constants $C^\star, d^\star$ introduced in (3.6). For the sake of simplicity we use a single $d^\star$, but note that the adaptive form of (3.9) allows to capture general cases like the one presented in Example 3.3, where different precision scales are characterized by different $d^\star$. The following corollary is proved in Section 3.5.3 and can be naturally extended to the case of different $d^\star$ by substituting the correct $d^\star$ when property 3.6 is invoked at the second-to-last inequality in the proof of Corollary 3.1.

Corollary 3.1. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $n \in \mathbb{N}^\star$, $\alpha \in [0, \varepsilon/25]$, $x_1 \in D$, and assume that the Piyavskii–Shubert* *algorithm (Algorithm 10) is run with inputs L_1, n, α, x_1. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6). Defining*

$$\overline{n} := 1 + C^\star \left(1 + 5.5\frac{L_1}{L_0}\mathbb{I}_{L_1 \neq L_0 \vee \alpha \neq 0}\right)^d \times \begin{cases} \log_2\left(\dfrac{\varepsilon_0}{\varepsilon}\right) + \log_2(1.08) & \text{if } d^\star = 0 \\ \dfrac{(1.08)^{d^\star}\left(\dfrac{\varepsilon_0}{\varepsilon}\right)^{d^\star} - 1}{2^{d^\star} - 1} & \text{if } d^\star > 0 \end{cases} \tag{3.10}$$

if $n \geqslant \overline{n}$, then the simple regret (3.1) of the Piyavskii–Shubert algorithm satisfies $r_n \leqslant \varepsilon$.

The analysis in Section 3.5.3 shows that the multiplicative term $\big(1 + (5.5)\varepsilon\frac{L_1}{L_0}\mathbb{I}_{L_1 \neq L_0 \vee \alpha \neq 0}\big)^d$ appears applying Lemma 3.6 (Section 3.5.1) because of the imperfect information on L_0 and the perturbations in the evaluations of f. If $L_0 = L_1$ and $\alpha = 0$ the term disappears. This suggests a quite striking discontinuity in the hardness of the problem with respect to the information available to the learner. A very accurate, but not exact, estimate of L_0 or evaluation of f still increases the sample complexity by a term exponential in the dimension of the ambient space.

We conclude the section by upper bounding the regret of the Piyavskii–Shubert algorithm when the values of the function are observed exactly. This is obtained directly by letting $\alpha = 0$ and solving Equation (3.10) for ε.

Corollary 3.2. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $n \in \mathbb{N}^\star$, $\alpha = 0$, $x_1 \in D$, and assume that the Piyavskii–Shubert algorithm (Algorithm 10) is run with inputs $L_1, n, 0, x_1$. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6). Then the simple regret (3.1) of the Piyavskii–Shubert algorithm satisfies*

$$r_n \leqslant \begin{cases} \exp\big(-\Omega(n)\big) & \text{if } d^\star = 0 \\ \mathcal{O}\big(n^{-1/d^\star}\big) & \text{if } d^\star > 0 \end{cases}$$

Proof of Theorem 3.1

In this section we present a formal proof of Theorem 3.1. We begin by proving an important property of the proxy function $\widehat{f}_k$. When f is globally Lipschitz, $\alpha = 0$, and D is compact, the lower bound $\widehat{f}_k(x) \geqslant f(x)$ is well known and true not only for $x = x^\star$ but for all $k \geqslant 1$ and all $x \in D$.

Lemma 3.1. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also $n \in \mathbb{N}^\star$, $\alpha \geqslant 0$, $x_1 \in D$, and assume that the Piyavskii–Shubert algorithm (Algorithm 10) is run with inputs L_1, n, α, x_1. Then, for all $k \geqslant 1$, the proxy function $\widehat{f}_k(x) = \min_{1 \leqslant i \leqslant k}\big\{f(x_i) + L_1\|x_i - x\| + (\xi_i + \alpha)\big\}$ is L_1-Lipschitz and satisfies, for all $j \in \{k, \dots, n\}$,*

$$\widehat{f}_k(x^\star) \geqslant f(x^\star) \quad \text{and} \quad \widehat{f}_j(x_k) \leqslant f(x_k) + 2\alpha\ .$$

Proof. Fix any $k \geqslant 1$ and $j \geqslant k$. The fact that $\widehat{f}_k$ is L_1-Lipschitz is straightforward. Moreover,

$$\widehat{f}_k(x^\star) = \min_{i \in \{1,\dots,k\}}\big\{f(x_i) + L_1\|x_i - x^\star\| + (\xi_i + \alpha)\big\}\ . \tag{3.11}$$

Since f is L_0-locally Lipschitz around $x^\star$ and $L_1 \geqslant L_0$, we get for all $i \in \{1, \dots, k\}$

$$f(x_i) \geqslant f(x^\star) - L_0\|x_i - x^\star\| \geqslant f(x^\star) - L_1\|x_i - x^\star\|\ .$$

Plugging $f(x_i) + L_1\|x_i - x^\star\| \geqslant f(x^\star)$ and $\xi_i \geqslant -\alpha$ in Equation (3.11) gives $\widehat{f}_k(x^\star) \geqslant f(x^\star)$. Furthermore, the definition of $\widehat{f}_j(x_k)$ and $\xi_k \leqslant \alpha$ imply, since $j \geqslant k$

$$\widehat{f}_j(x_k) = \min_{i \in \{1,\dots,j\}}\big\{f(x_i) + L_1\|x_i - x_k\| + (\xi_i + \alpha)\big\} \leqslant f(x_k) + L_1\|x_k - x_k\| + (\xi_k + \alpha) \leqslant f(x_k) + 2\alpha$$

which concludes the proof. □

Before proving the main result, we first state a useful lemma which shows that if the Piyavskii–Shubert algorithm observes f at a Δ-suboptimal point x_i, then the next query points x_j are all distant from x_i by at least $\Omega(\Delta)$. In other words, the Piyavskii–Shubert algorithm does not explore too much in suboptimal regions.

Lemma 3.2. *Suppose that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also $n \in \mathbb{N}^\star$, $\alpha \geqslant 0$, $x_1 \in D$, and assume that the Piyavskii–Shubert algorithm (Algorithm 10) is run with inputs L_1, n, α, x_1. Fix any $\Delta > 0$ and assume that there exists $i \in \{1, \dots, n-1\}$ such that the point x_i queried by the Piyavskii–Shubert algorithm during the i-th iteration satisfies $x_i \in \mathcal{X}_\Delta^c$. Then, for all $j > i$, the j-the queried point satisfies*

$$\|x_j - x_i\| > \frac{\Delta - 3\alpha}{L_1} \, .$$

Proof. Assume that $x_i \in \mathcal{X}_\Delta^c$ for some $i \geqslant 1$ and let $j > i$. Then

$$\begin{aligned}
\widehat{f}_{j-1}(x_j) &\geqslant \widehat{f}_{j-1}(x^\star) - \alpha && (x_j \text{ was selected at iteration } j) \\
&\geqslant f(x^\star) - \alpha && (\text{by Lemma 3.1}) \\
&> f(x_i) + \Delta - \alpha && (x_i \in \mathcal{X}_\Delta^c) \\
&\geqslant \widehat{f}_{j-1}(x_i) + \Delta - 3\alpha && (\text{by Lemma 3.1})
\end{aligned}$$

Since $\widehat{f}_{j-1}$ is L_1-Lipschitz (Lemma 3.1), we have $L_1\|x_j - x_i\| \geqslant |\widehat{f}_{j-1}(x_j) - \widehat{f}_{j-1}(x_i)| > \Delta - 3\alpha$. □

We can now prove Theorem 3.1, the main result of Section 3.3.1. To this end, we use a peeling technique in which the input space D is partitioned in terms of the output values of f.

Proof of Theorem 3.1. Let $m_\varepsilon := \lceil \log_2(\varepsilon_0 \varepsilon^{-1}) \rceil \geqslant 1$. We use a peeling technique and partition the set $\mathcal{X}_\varepsilon^c$ of ε-suboptimal points into multiple layers (recall Equation (3.4)). Note that

$$(\varepsilon_0 2^{-m_\varepsilon}, +\infty) = (\varepsilon_0 2^{-m_\varepsilon}, \varepsilon_0 2^{-m_\varepsilon+1}] \cup \dots \cup (\varepsilon_0/2, \varepsilon_0] \cup (\varepsilon_0, +\infty) \, .$$

Moreover $\mathcal{X}_{\varepsilon_0 2^{-m_\varepsilon}} \subset \mathcal{X}_\varepsilon$ and by definition of ε_0 (Equation (3.5)), $\mathcal{X}_{\varepsilon_0}^c = \varnothing$. Thus we have

$$\mathcal{X}_\varepsilon^c \subset \mathcal{X}_{\varepsilon_0 2^{-m_\varepsilon}}^c \subset \left(\bigcup_{s=1}^{m_\varepsilon} A_s \right)$$

with $A_s := \mathcal{X}_{(\varepsilon_0 2^{-s}, \varepsilon_0 2^{-s+1}]}$ for all $1 \leqslant s \leqslant m_\varepsilon$. We remark that $\{A_s\}_{1 \leqslant s \leqslant m_\varepsilon}$ is a collection of disjoint subsets of D. Therefore, for any $1 \leqslant k \leqslant n$, if $x_k \in \mathcal{X}_\varepsilon^c$, there exists a unique $s \in \{1, \dots, m_\varepsilon\}$ such that $x_k \in A_s$. For any $s \in \{1, \dots, m_\varepsilon\}$, by Lemma 3.2, the maximum number of rounds k at which x_k can be chosen in $A_s = \mathcal{X}_{\varepsilon_0 2^{-s}}^c \cap \mathcal{X}_{\varepsilon_0 2^{-s+1}}$ is upper bounded by $\mathcal{N}\left(A_s, \frac{\varepsilon_0 2^{-s} - 3\alpha}{L_1}\right)$. Then

$$\left| \left\{ k \in \{1, \dots, n\} : x_k \in \mathcal{X}_\varepsilon^c \right\} \right| \leqslant \sum_{s=1}^{m_\varepsilon} \left| \left\{ k \in \{1, \dots, n\} : x_k \in A_s \right\} \right| \leqslant \sum_{s=1}^{m_\varepsilon} \mathcal{N}\left(A_s, \frac{\varepsilon_0 2^{-s} - 3\alpha}{L_1}\right) \, .$$

Therefore, if

$$n \geqslant \sum_{s=1}^{m_\varepsilon} \mathcal{N}\left(A_s, \frac{\varepsilon_0 2^{-s} - 3\alpha}{L_1}\right) + 1$$

then there exists $k \in \{1, \dots, n\}$ such that $x_k \in \mathcal{X}_\varepsilon$, which implies (by definition of $x_n^\star$) that

$$f(x_n^\star) \geqslant f(x_k) + \xi_k - \xi_{i_n^\star} \geqslant f(x^\star) - 2\alpha - \varepsilon \, .$$

□

3.3.2 Automatic stopping

We now study a version of the Piyavskii–Shubert algorithm that stops automatically outputting a near-optimal solution (Algorithm 11) after stopping. We derive an upper bound on the number n of queries to f needed to automatically stop (Theorem 3.2 and Corollary 3.3). Formally, the algorithm applies to the following online optimization protocol.

A function $f\colon D \subset \mathbb{R}^d \to \mathbb{R}$ that attains its maximum at some point $x^\star \in D$ is fixed in advance. Only the domain D of the function, an upper bound $\alpha \geqslant 0$ on the absolute value of the perturbations (see below), and a prescribed accuracy $\varepsilon > 0$ are known to the learner in advance.

For each $k = 1, 2, \ldots$:

1. the learner chooses a point $x_k \in D$,
2. the environment picks a deterministic perturbation
$$\xi_k \in [-\alpha, \alpha] \,, \tag{3.12}$$
depending on f and all points x_s chosen by the learner up to and including the current one x_k,
3. the value $f(x_k) + \xi_k$ is revealed to the learner.

At the end of each iteration n the learner can decide to interrupt the process and output a prediction $x_n^\star$ with the goal of returning an ε-optimal point. In other words, the regret $r_n := f(x^\star) - f(x_n^\star)$ at time n has to satisfy
$$r_n \leqslant \varepsilon \,. \tag{3.13}$$

The ε-Piyavskii–Shubert algorithm (Algorithm 11, below) applies to our non-compact d-dimensional setting with perturbations. It behaves similarly to Algorithm 10. A proxy $\widehat{f}_k$ of the objective function f that (up to perturbations) upper bounds f at least at the maximizer $x^\star$ is maintained and updated during each iteration. After each new evaluation, the graph of the proxy gets closer and closer to that of the objective. The points that are queried by the algorithm are those where the $\widehat{f}_k$ is biggest. The intuition is that if $\widehat{f}_k$ is sufficiently close to f, then the point with the highest value among all proxy functions should also be near-optimal for f. The stopping rule at the beginning of the loop triggers as soon as the maximum of the observed values gets close to the maximum of all proxy function. At a high level, when these two numbers get close, it means that the proxy function became sufficiently good of an approximation of the objective.

Algorithm 11: ε-Piyavskii–Shubert algorithm (with known accuracy ε)

```
  Input: Lipschitz constant $L_1$, accuracy $\varepsilon > 0$, perturbation scale $\alpha > 0$, initial guess $x_1 \in D$
  Initialization: $k \leftarrow 0$ and $\widehat{f}_0^\star - f_0^\star \leftarrow \varepsilon + 2\alpha + 1$
1 while $\widehat{f}_k^\star - f_k^\star > \varepsilon + 2\alpha$ do
2     update $k \leftarrow k + 1$;
3     observe $y_k \leftarrow f(x_k) + \xi_k$;
4     update $\widehat{f}_k(x) \leftarrow \min_{i \in \{1,\ldots,k\}} \bigl\{ y_i + L_1 \|x_i - x\| + \alpha \bigr\}$ for all $x \in D$;
5     pick $x_{k+1} \in D$ such that $\widehat{f}_k(x_{k+1}) \geqslant \sup_{x \in D} \widehat{f}_k(x) - \alpha$;
6     update $\widehat{f}_k^\star \leftarrow \widehat{f}_k(x_{k+1})$ and $f_k^\star \leftarrow \max_{i \in \{1,\ldots,k\}} y_i$;
7 return $x_k^\star \leftarrow x_{i_k^\star}$;
```

Note that (similarly to Algorithm 10) in Algorithm 11, we pick x_{k+1} to be an α-optimal point of $\widehat{f}_k$. The algorithm could be defined slightly more generally by picking x_{k+1} as an η-optimal point of $\widehat{f}_k$, where η is an additional parameter independent of the bound α on $|\xi_k|$. Our choice to lighten the notation is due to the fact that if $\alpha > 0$, in order to have a meaningful result, η should eventually be set to $\mathcal{O}(\alpha)$ anyway.

Moreover, with our choice, the case $\alpha = 0$ still includes the classic setting in which D is compact and values are observed with no perturbations.

The ε-Piyavskii–Shubert algorithm (Algorithm 11) is an adaptive version of the Piyavskii–Shubert algorithm (Algorithm 10). The main result of this section (Theorem 3.2) shows that the former stops after roughly the same number of iterations as the latter, but it does not rely on any prior knowledge of $\widetilde{n}$ (defined in Equation (3.9)) nor its upper bound $\overline{n}$ (defined in Equation (3.10)), which depends on the possibly unknown constants $d^\star$ and $C^\star$. Nevertheless, it still returns an ε-optimal point after stopping.

The ε-Piyavskii–Shubert algorithm was extensively studied in dimension $d = 1$ when $\alpha = 0$. For example, Hansen et al. [40] derived a tight upper bound on the number of evaluations of f needed before stopping, in terms of an integral involving the increments $f(x^\star) - f(x)$. In Section 3.5.4 we discuss how to derive a regret bound similar to that of Corollary 3.3 from their upper bound. Our result could thus be interpreted as a generalization of this result to an arbitrary dimension $d \geqslant 1$.

We now state the main result of this section, which we prove in Section 7.

Theorem 3.2. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $\alpha \in [0, \varepsilon/24)$, $x_1 \in D$, and assume that the ε-Piyavskii–Shubert algorithm (Algorithm 11) is run with inputs $L_1, \varepsilon, \alpha, x_1$. Then, the ε-Piyavskii–Shubert algorithm stops after $n \leqslant \widetilde{n}'$ iterations, where $\widetilde{n}'$ is defined by*

$$\widetilde{n}' := \mathcal{N}\left(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon - \alpha}{L_1}\right) + \sum_{s=0}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil} \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-s-1}, \varepsilon_0 2^{-s}]}, \frac{\varepsilon_0 2^{-s-1} - 3\alpha}{L_1}\right) + 1 \tag{3.14}$$

and its simple regret (3.1) satisfies $r_n \leqslant \varepsilon + 2\alpha$.

Note that the definition of $\widetilde{n}'$ (Equation (3.14)) is remarkably similar to that of $\widetilde{n}$ (Equation (3.9)). The price of adapting is quantified by the two terms in the difference $\widetilde{n}' - \widetilde{n}$. The first of the two terms, $\mathcal{N}\big(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon-\alpha}{L_1}\big)$, is particularly important, as it reveals a subtle but crucial difference between the Piyavskii–Shubert algorithm (Algorithm 10) and the ε-Piyavskii–Shubert algorithm (Algorithm 11). Consider a constant function f. Since all suboptimal layers are empty in this case (because all points in D are optimal), Equation (3.9) reflects the fact that as little as $\widetilde{n} = 1$ evaluation is needed in order for the Piyavskii–Shubert algorithm (Algorithm 10) to guarantee a near-optimal solution, as the first prediction is necessarily already optimal. However, with the same constant objective, the number of evaluations $\widetilde{n}'$ needed for the ε-Piyavskii–Shubert algorithm (Algorithm 11) to stop is of order $\mathcal{N}\big(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon-\alpha}{L_1}\big) \gtrsim \big(\frac{1}{\varepsilon}\big)^d$, which is as big as it gets! This huge gap is unavoidable and due to the fact that Algorithm 11 is asked to complete a task that is significantly harder than that of Algorithm 10. The Piyavskii–Shubert algorithm simply runs for a prescribed amount of iterations, where it happens to make good predictions if the objective is flat. In contrast, the ε-Piyavskii–Shubert algorithm has to be sure that when it stops, it outputs a good prediction. If the function is very flat, a full grid search has to be performed in order to make sure that the proxy function at his highest value is close enough to the value of the objective.

As we did in the previous section, we now express the previous result in terms of the constants $C^\star, d^\star$ introduced in (3.6). The following corollary is proved in Section 3.5.3.

Corollary 3.3. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $\alpha \in [0, \varepsilon/25]$, $x_1 \in D$, and assume that the ε-Piyavskii–Shubert algorithm (Algorithm 11) is run with inputs $L_1, \varepsilon, \alpha, x_1$. Fix any $C^\star \geqslant 0$ and $d^\star \in [0, d]$ satisfying (3.6). Then, the ε-Piyavskii–Shubert algorithm stops after $n \leqslant \overline{n}'$ iterations, where $\overline{n}'$ is defined by*

$$\overline{n}' := 1 + C^\star \left(1 + 15.5 \frac{L_1}{L_0} \mathbb{I}_{L_1 \neq L_0 \vee \alpha \neq 0}\right)^d \times \begin{cases} \log_2\left(\dfrac{\varepsilon_0}{\varepsilon}\right) + \log_2(1.08) + 3 & \text{if } d^\star = 0 \\[2ex] \dfrac{\left(2^{d^\star} + 2^{d^\star+1} - 2\right)(1.08)^{d^\star}\left(\dfrac{\varepsilon_0}{\varepsilon}\right)^{d^\star} - 1}{2^{d^\star} - 1} & \text{if } d^\star > 0 \end{cases}$$

and its simple regret (3.1) satisfies $r_n \leqslant \varepsilon$.

Proof of Theorem 3.2

In this section we present a formal proof of Theorem 3.2. Before proving the main result, we first state a useful lemma analogous to Lemma 3.2. It shows that if the ε-Piyavskii–Shubert algorithm observes f at a suboptimal point x_i, then the next query points x_j are all distant from x_i and in any case it never queries points that are too close to each other. In other words, the ε-Piyavskii–Shubert algorithm does not explore too much in suboptimal regions and it does not waste evaluations on neighboring points.

Lemma 3.3. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1) and let $L_1 \geqslant L_0$. Let also $\varepsilon \in (0, \varepsilon_0)$, $\alpha \geqslant 0$, $x_1 \in D$, and assume that the ε-Piyavskii–Shubert algorithm (Algorithm 11) is run with inputs $L_1, \varepsilon, \alpha, x_1$. Assume that the ε-Piyavskii–Shubert algorithm runs for at least $j > 1$ iterations and let $1 \leqslant i < j$. Then*

$$\|x_i - x_j\| > \frac{\varepsilon - \alpha}{L_1} \, .$$

Moreover, if there exists $0 \leqslant k \leqslant m_\varepsilon := \lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil$ such that $x_i \in \mathcal{X}_{(\varepsilon_0 2^{-k-1}, \varepsilon_0 2^{-k}]}$, then

$$\|x_i - x_j\| > \frac{\varepsilon_0 2^{-k-1} - 3\alpha}{L_1} \, .$$

Proof of Lemma 3.3. Note that

$$\begin{aligned}
\widehat{f}_{j-1}(x_j) &= \widehat{f}^\star_{j-1} && \text{(by definition of } \widehat{f}^\star_{j-1}\text{)} \\
&> f^\star_{j-1} + \varepsilon + 2\alpha && \text{(the algorithm ran for at least } j \text{ iterations)} \\
&\geqslant f(x_i) + \varepsilon + \alpha && \text{(by definition of } f^\star \text{ and } \xi_i \geqslant -\alpha\text{)} \\
&\geqslant \widehat{f}_{j-1}(x_i) + \varepsilon - \alpha \, . && \text{(by Lemma 3.1)}
\end{aligned}$$

Since $\widehat{f}_{j-1}$ is L_1-Lipschitz (Lemma 3.1), we have $L_1 \|x_i - x_j\| \geqslant \big|\widehat{f}_{j-1}(x_j) - \widehat{f}_{j-1}(x_i)\big| > \varepsilon - \alpha$, which gives the first inequality. For the second inequality, we proceed as in the proof of in Lemma 3.2. Let $\Delta = \varepsilon_0 2^{-k-1}$ and note that $x_i \in \mathcal{X}^{\mathrm{c}}_\Delta$. Then

$$\begin{aligned}
\widehat{f}_{j-1}(x_j) &\geqslant \widehat{f}_{j-1}(x^\star) - \alpha && \text{(by definition of } x_j\text{)} \\
&\geqslant f(x^\star) - \alpha && \text{(by Lemma 3.1)} \\
&> f(x_i) + \Delta - \alpha && (x_i \in \mathcal{X}^{\mathrm{c}}_\Delta) \\
&\geqslant \widehat{f}_{j-1}(x_i) + \Delta - 3\alpha \, . && \text{(by Lemma 3.1)}
\end{aligned}$$

Since $\widehat{f}_{j-1}$ is again L_1-Lipschitz (Lemma 3.1), we have $L_1 \|x_j - x_i\| \geqslant |\widehat{f}_{j-1}(x_j) - \widehat{f}_{j-1}(x_i)| > \Delta - 3\alpha$, which concludes the proof. $\square$

We can now prove Theorem 3.2, the main result of the deterministic setting. The proof proceeds similarly to that of Theorem 3.1: we rely on a peeling technique in which the input space D is partitioned in terms of the output values of f.

Proof of Theorem 3.2. Fix any $\varepsilon \in (0, \varepsilon_0)$ and $\alpha \in [0, \varepsilon/24)$. Let $\mathcal{T} \subset \mathbb{N}^*$ be the set of indices of all iterations performed by the ε-Piyavskii–Shubert algorithm (Algorithm 11) before stopping, $m_\varepsilon := \lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil \geqslant 1$, and for all $k \in \{0, \dots, m_\varepsilon\}$,

$$\begin{aligned}
\mathcal{T}_k &:= \left\{ t \in \mathcal{T} : x_t \in \mathcal{X}_{(\varepsilon_0 2^{-k-1}, \varepsilon_0 2^{-k}]} \right\} \, , \\
\mathcal{T}_{m_\varepsilon + 1} &:= \left\{ t \in \mathcal{T} : x_t \in \mathcal{X}_{\varepsilon/2} \right\} \, .
\end{aligned}$$

Lemma 3.3 implies, for all $k \in \{0, \dots, m_\varepsilon\}$,

$$|\mathcal{T}_k| \leqslant \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-k-1}, \varepsilon_0 2^{-k}]}, \frac{\varepsilon_0 2^{-k-1} - 3\alpha}{L_1}\right) ,$$
$$|\mathcal{T}_{m_\varepsilon+1}| \leqslant \mathcal{N}\left(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon - \alpha}{L_1}\right) .$$

Since $\mathcal{X}_{(\varepsilon_0, +\infty)} = \varnothing$ (by definition of ε_0 in Equation (3.5)) and $[0, \varepsilon_0] = [0, \varepsilon/2] \cup (\varepsilon_0 2^{-m_\varepsilon - 1}, \varepsilon_0 2^{-m_\varepsilon}] \cup \dots \cup (\varepsilon_0 2^{-1}, \varepsilon_0]$, we have $D = \mathcal{X}_{\varepsilon/2} \cup \left(\bigcup_{s=0}^{m_\varepsilon} \mathcal{X}_{(\varepsilon_0 2^{-1-s}, \varepsilon_0 2^{-s}]}\right)$. Hence, for all iterations $i \in \mathcal{T}$ there exists $k \in \{0, \dots, m_\varepsilon + 1\}$ such that $i \in \mathcal{T}_k$, which in turn gives

$$|\mathcal{T}| \leqslant \sum_{k=0}^{m_\varepsilon+1} |\mathcal{T}_k| \leqslant \mathcal{N}\left(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon - \alpha}{L_1}\right) + \sum_{k=0}^{m_\varepsilon} \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-k-1}, \varepsilon_0 2^{-k}]}, \frac{\varepsilon_0 2^{-k-1} - 3\alpha}{L_1}\right) .$$

□

3.4 Stochastic perturbations

In this section we show how to apply the results proven in Section 3.3 to an algorithm designed for the stochastic setting (Algorithm 12). More precisely, we assume that the values of f are observed up to subgaussian noise. We then use a mini-batch sampling technique to produce tight estimates of the value of f at each iteration of the algorithm.

Assumption 3.2. *Let $\xi := (\xi_{k,i})_{k,i \in \mathbb{N}^*}$ be a sequence of independent random variables. We assume that there exists $\sigma_0 \geqslant 0$ such that all random variables in the sequence are σ_0-subgaussian, i.e., for all $i, k \in \mathbb{N}^*$ the random variable $\xi_{k,i}$ is centered and for all $\lambda \in \mathbb{R}$,*

$$\mathbb{E}\left(e^{\lambda \xi_{k,i}}\right) \leqslant e^{\lambda^2 \sigma_0^2 / 2} .$$

We say that ξ is a σ_0-subgaussian noise sequence. Moreover, we assume that an upper bound σ_1 on σ_0 is known to the learner.

We study directly the version of the problem in which an accuracy level ε is given and the algorithm stops automatically guaranteeing an ε-optimal solution. Formally, the algorithm applies to the following online optimization protocol.

A function $f\colon D \subset \mathbb{R}^d \to \mathbb{R}$ that attains its maximum at some point $x^\star \in D$ and a σ_0-subgaussian noise sequence ξ (Assumption 3.2) are fixed in advance. Only the domain D of the function, an upper bound σ_1 on σ_0, and a prescribed accuracy $\varepsilon > 0$ are known to the learner in advance.

For each $k = 1, 2, \dots$:

1. the learner queries a point $x_k \in D$ for a finite amount of times,
2. for each query i of x_k, the value $f(x_k) + \xi_{k,i}$ is revealed to the learner.

At the end of each iteration n the learner can decide to interrupt the process and output a prediction $x_n^\star$ with the goal of returning an ε-optimal point. In other words, the regret $r_n := f(x^\star) - f(x_n^\star)$ at time n has to satisfy

$$r_n \leqslant \varepsilon . \tag{3.15}$$

The stochastic ε-Piyavskii–Shubert algorithm (Algorithm 12) applies to our non-compact d-dimensional setting with stochastic perturbations. It behaves similarly to Algorithm 11. We maintain a proxy $\widehat{f}_k$ of the objective function f that (up to perturbations) upper bounds f at least at the maximizer $x^\star$. The proxy is updated during each iteration by querying the same value multiple times in order to build an estimate of the real value of the function at that point. After each update, the graph of the proxy gets closer and closer to that of the objective. The points that are queried by the algorithm are those where the $\widehat{f}_k$ is biggest. The intuition is that if $\widehat{f}_k$ is sufficiently close to f, then the point with the highest value among all proxy functions should also be near-optimal for f. The stopping rule at the beginning of the loop triggers as soon as the maximum of the observed values gets close to the maximum of all proxy function. At a high level, when these two numbers get close, it means that the proxy function became sufficiently good of an approximation of the objective.

Algorithm 12: Stochastic ε-Piyavskii–Shubert algorithm

```
Input: Lipschitz constant $L_1$, subgaussian constant $\sigma_1 > 0$, accuracy $\varepsilon > 0$, confidence
       $1-\delta \in (0,1)$, initial guess $x_1 \in D$.
Initialization: $\varepsilon' \leftarrow, (25/27)\varepsilon$, $\alpha \leftarrow \varepsilon/27$, $k \leftarrow 0$ and $\widehat{f}_0^\star - f_0^\star \leftarrow \varepsilon' + 2\alpha + 1$
1 while $\widehat{f}_k^\star - f_k^\star > \varepsilon' + 2\alpha$ do
2     update $k \leftarrow k+1$ and $m_k = \left\lceil \frac{2\sigma_1^2}{\alpha^2} \ln\left(\frac{2k(k+1)}{\delta}\right) \right\rceil$;
3     observe $f(x_k) + \xi_{k,1}, \dots, f(x_k) + \xi_{k,m_k}$ and compute $y_k \leftarrow f(x_k) + \overline{\xi}_k$, where $\overline{\xi}_k = \frac{1}{m_k}\sum_{i=1}^{m_k} \xi_{k,i}$;
4     update $\widehat{f}_k(x) \leftarrow \min_{i\in\{1,\dots,k\}} \{ y_i + L_1 \lVert x_i - x \rVert + \alpha \}$ for all $x \in D$;
5     pick $i_k^\star \in \arg\max_{i\in\{1,\dots,k\}} y_i$ and $x_{k+1} \in D$ such that $\widehat{f}_k(x_{k+1}) \geqslant \sup_{x\in D} \widehat{f}_k(x) - \alpha$;
6     update $f_k^\star \leftarrow f\big(x_{i_k^\star}\big) + \overline{\xi}_{i_k^\star} = \max_{i\in\{1,\dots,k\}} y_i$ and $\widehat{f}_k^\star \leftarrow \widehat{f}_k(x_{k+1}) \geqslant \sup \widehat{f}_k - \alpha$;
7 return $x_k^\star \leftarrow x_{i_k^\star}$
```

Before proving the main result of the section, we state a well-known concentration lemma for the empirical average of subgaussian random variables (see, e.g., [12, Section 2.3]).

Lemma 3.4. *Let ξ be a σ_0-subgaussian noise sequence (Assumption 3.2). Then, for all $k, m \in \mathbb{N}$ and all $\alpha \geqslant 0$,*

$$\mathbb{P}\left(\left| \frac{1}{m} \sum_{i=1}^{m} \xi_{k,i} \right| \geqslant \alpha \right) \leqslant 2e^{-m\alpha^2/(2\sigma_0^2)} \leqslant 2e^{-m\alpha^2/(2\sigma_1^2)} . \tag{3.16}$$

Similarly to the previous section, we upper bound the number of evaluations of f after which the stochastic ε-Piyavskii–Shubert algorithm (Algorithm 12) automatically stops outputting an ε-optimal point. The proof of the following theorem is a composition of the analysis of Theorem 3.2 (Section 7) and a carefully designed use of concentration inequalities.

Theorem 3.3. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1), the noise ξ is σ_0-subgaussian (Assumption 3.2), and let $L_1 \geqslant L_0$ and $\sigma_1 \geqslant \sigma_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $\delta \in (0,1)$, $x_1 \in D$, and assume that the stochastic ε-Piyavskii–Shubert algorithm (Algorithm 12) is run with inputs $L_1, \sigma_1, \varepsilon, 1-\delta, x_1$. Then, with probability at least $1-\delta$, the stochastic ε-Piyavskii–Shubert algorithm stops after performing $N \leqslant \widetilde{N}'$ evaluations of f, where*

$$\widetilde{N}' := 2916 \frac{\sigma_1^2}{\varepsilon^2} \big(\widetilde{n}' + 1\big) \ln\left(\frac{4\big(\widetilde{n}' + 1\big)}{\delta} \right) ,$$

$$\widetilde{n}' := \mathcal{N}\left(\mathcal{X}_{\frac{25}{54}\varepsilon}, \frac{24}{27}\frac{\varepsilon}{L_1} \right) + \sum_{k=0}^{\left\lceil \log_2\left(\frac{27}{25}\frac{\varepsilon_0}{\varepsilon}\right) \right\rceil} \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-k-1}, \varepsilon_0 2^{-k}]}, \frac{\varepsilon_0 2^{-k-1} - \varepsilon/9}{L_1} \right) ,$$

and its simple regret (3.1) satisfies $r_N \leqslant \varepsilon$.

Note again the presence of the first term in the definition of $\widetilde{n}'$. As highlighted before (see discussion after Theorem 3.2), this additional term is due to the price of automatic stopping and cannot be avoided.

Proof. Let $\varepsilon' = (25/27)\varepsilon$ and $\alpha = \varepsilon/27$ as in the initialization of the stochastic ε-Piyavskii–Shubert algorithm (Algorithm 12). Consider the "bad" event $E := \{\exists k \in \mathbb{N}^* : |\overline{\xi}_k| > \alpha\}$. By applying a union bound, Inequality (3.16), and the definition of $m_k = \lceil (2\sigma_1^2/\alpha^2)\ln\bigl(2k(k+1)/\delta\bigr)\rceil$, we get

$$\mathbb{P}(E) \leqslant \sum_{k\in\mathbb{N}^*} \frac{\delta}{k(k+1)} = \delta \,.$$

For each outcome belonging to the complement of E, during each iteration k the value $f(x_k)$ is observed up to a perturbation $\overline{\xi}_k$ with $|\overline{\xi}_k| \leqslant \alpha$. With probability at least $1-\delta$ we can therefore apply Theorem 3.2 to upper bound the number of iterations before stopping by $\widetilde{n}'$, which in turn gives that the total number of evaluations of f before stopping is at most

$$\sum_{i=1}^{\widetilde{n}'} m_i \leqslant \sum_{i=1}^{\widetilde{n}'} \left(\frac{4\sigma_1^2}{\alpha^2}\ln(i) + \frac{2\sigma_1^2}{\alpha^2}\ln\left(\frac{4}{\delta}\right) + 1 \right) \leqslant \frac{4\sigma_1^2}{\alpha^2}\bigl(\widetilde{n}'+1\bigr)\ln\left(\frac{4\bigl(\widetilde{n}'+1\bigr)}{\delta}\right)$$

where in the last inequalities we used $2k(k+1) \leqslant 4k^2$ and

$$\sum_{k=1}^{n} \ln(k) \leqslant \int_1^{n+1} \ln(x)\,\mathrm{d}x = (n+1)\ln(n+1) - n \,.$$

□

Combining the proofs of Corollary 3.1 and Theorem 3.3 gives immediately the following upper bound on the number of evaluations of f after which the stochastic ε-Piyavskii–Shubert algorithm (Algorithm 12) automatically stops.

Corollary 3.4. *Assume that f is L_0-Lipschitz around $x^\star$ (Assumption 3.1), the noise ξ is σ_0-subgaussian (Assumption 3.2), and let $L_1 \geqslant L_0$ and $\sigma_1 \geqslant \sigma_0$. Let also ε_0 be defined by Equation (3.5), $\varepsilon \in (0, \varepsilon_0)$, $\delta \in (0,1)$, $x_1 \in D$, and assume that the stochastic ε-Piyavskii–Shubert algorithm (Algorithm 12) is run with inputs $L_1, \sigma_1, \varepsilon, 1-\delta, x_1$. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6). Then, with probability at least $1-\delta$, the stochastic ε-Piyavskii–Shubert algorithm stops after performing $N \leqslant \overline{N}'$ evaluations of f, where*

$$\overline{N}' := 2916\frac{\sigma_1^2}{\varepsilon^2}\bigl(\overline{n}'+1\bigr)\ln\left(\frac{4\bigl(\overline{n}'+1\bigr)}{\delta}\right) ,$$

$$\overline{n}' := 1 + C^\star\left(1 + 8.5\frac{L_1}{L_0}\right)^d \times \begin{cases} \log_2\left(\dfrac{\varepsilon_0}{\varepsilon}\right) + \log_2(1.08) + 3 & \textit{if } d^\star = 0 \\[2ex] \dfrac{\left(2^{d^\star} + 2^{d^\star+1} - 2\right)(1.08)^{d^\star}\left(\dfrac{\varepsilon_0}{\varepsilon}\right)^{d^\star} - 1}{2^{d^\star} - 1} & \textit{if } d^\star > 0 \end{cases}$$

and its simple regret (3.1) satisfies $r_N \leqslant \varepsilon$.

3.5 Deferred proofs and additional results

3.5.1 Useful inequalities about packing and covering numbers

Covering numbers and packing numbers (see Section 3.2.2) are closely related. In particular, the following well-known inequalities hold—see, e.g., [93, Lemma 5.5 and Example 5.8, with permuted notation of $\mathcal{M}$ and $\mathcal{N}$].[4]

[4]The definition of r-covering number of a subset A of $\mathbb{R}^d$ implied by [93, Definition 5.1] is slightly stronger than the one used in this chapter, because elements $x_1, \ldots, x_N$ of r-covers belong to A rather than just $\mathbb{R}^d$. Even if we do not need it for our analysis, Inequality (3.18) holds also in this stronger sense.

Lemma 3.5. *For any nonempty bounded set $A \subset \mathbb{R}^d$ and any real number $r > 0$,*

$$\mathcal{N}(A, 2r) \leqslant \mathcal{M}(A, r) \leqslant \mathcal{N}(A, r) \ . \tag{3.17}$$

Furthermore, for all $\delta > 0$ and all $r > 0$,

$$\mathcal{M}\left(B_{\|\cdot\|}(\delta), r\right) \leqslant \left(1 + 2\frac{\delta}{r}\mathbb{I}_{r<\delta}\right)^d \ . \tag{3.18}$$

We now state a known lemma about packing numbers at different scales. This result will be useful to control how an overestimation L_1 of the Lipschitz constant L_0 impacts the regret of our algorithms.

Lemma 3.6. *For any nonempty bounded set $A \subset \mathbb{R}^d$ and any real numbers $r_1, r_2 > 0$,*

$$\mathcal{N}(A, r_1) \leqslant \left(1 + 4\frac{r_2}{r_1}\mathbb{I}_{r_2>r_1}\right)^d \times \mathcal{N}(A, r_2) \ .$$

Proof. We can assume without loss of generality that $0 < r_1 < r_2$. Then,

$$\begin{aligned}
\mathcal{N}(A, r_1) &\leqslant \mathcal{M}(A, r_1/2) && \text{(by (3.17))} \\
&\leqslant \mathcal{M}(A, r_2) \times \mathcal{M}\big(B_{\|\cdot\|}(r_2), r_1/2\big) && \text{(see below)} \\
&\leqslant \mathcal{N}(A, r_2) \times \mathcal{M}\big(B_{\|\cdot\|}(r_2), r_1/2\big) && \text{(by (3.17))} \\
&\leqslant \mathcal{N}(A, r_2) \times \left(1 + \frac{4r_2}{r_1}\right)^d \ . && \text{(by (3.18))}
\end{aligned}$$

The second inequality is obtained by building the $r_1/2$-covering of A in two steps. First, we cover A with balls of radius r_2. Second, we cover each ball of the first cover with balls of radius $r_1/2$. □

3.5.2 Simple bound on the near-optimality dimension

The next well-known lemma shows that Inequality (3.6) is always true with $C^\star = 9^d$ and $d^\star = d$ (though significantly smaller values of $C^\star$ and $d^\star$ may exist, see Section 3.2.3). We recall that throughout the chapter $D \subset \mathbb{R}^d$ is bounded.

Lemma 3.7. *Assume that $f : D \to \mathbb{R}$ is L_0-Lipschitz around a maximizer $x^\star$ (Assumption 3.1) and set $\varepsilon_0 = L_0 \sup_{x,y \in D} \|x - y\|$. Then, for all $\varepsilon \in (0, \varepsilon_0]$,*

$$\mathcal{N}\left(\mathcal{X}_\varepsilon, \frac{\varepsilon}{2L_0}\right) \leqslant 9^d \left(\frac{\varepsilon_0}{\varepsilon}\right)^d \ .$$

Proof. Let $\varepsilon \in (0, \varepsilon_0]$. Fix any $a \in D$ and set $B = B_{\|\cdot\|}\left(\sup_{y\in D}\|y - a\|\right)$. Then $\mathcal{X}_\varepsilon \subset D \subset a + B$, which in turn yields

$$\mathcal{N}\left(\mathcal{X}_\varepsilon, \frac{\varepsilon}{2L_0}\right) \leqslant \mathcal{N}\left(a + B, \frac{\varepsilon}{2L_0}\right) \leqslant \mathcal{M}\left(B, \frac{\varepsilon}{4L_0}\right) \leqslant \left(1 + \frac{8\sup_{y\in D}\|y - a\|L_0}{\varepsilon}\right)^d \leqslant 9^d \left(\frac{\varepsilon_0}{\varepsilon}\right)^d \ ,$$

where the second inequality follows by translation invariance and by (3.17), the third by (3.18), and the last one by definition of ε_0. □

When the minimum

$$d^\star(L_0) := \min\left\{d' \in \mathbb{R}^+ : \exists C > 0, \forall \varepsilon \in (0, \varepsilon_0], \mathcal{N}\left(\mathcal{X}_\varepsilon, \frac{\varepsilon}{2L_0}\right) \leqslant C\left(\frac{\varepsilon_0}{\varepsilon}\right)^{d'}\right\}$$

exists, the quantity $d^\star(L_0)$ is called the *near-optimality dimension* of f [19]. The *zooming dimension* is defined similarly by packing the layers $\mathcal{X}_{(\varepsilon/2,\varepsilon]}$ instead of the sets $\mathcal{X}_\varepsilon$ (see [53]). When $d^\star(L_0)$ is well defined, Inequality (3.6) is satisfied with $d^\star = d^\star(L_0)$ and the constant $C^\star$ equal to

$$C^\star(L_0) := \min\left\{ C > 0 : \forall \varepsilon \in (0,\varepsilon_0], \mathcal{N}\left(\mathcal{X}_\varepsilon, \frac{\varepsilon}{2L_0}\right) \leqslant C\left(\frac{\varepsilon_0}{\varepsilon}\right)^{d^\star(L_0)} \right\} .$$

Note however that a small $d^\star(L_0)$ does not necessarily imply that the sets $\mathcal{X}_\varepsilon$ are small, since the constant $C^\star(L_0)$ can be arbitrarily large. Consider, for example, a small $\rho > 0$ and the function $x \mapsto \max\bigl\{\rho - \|x - x_0\|, 0\bigr\}$. Its near-optimality dimension is 0 but picking $d^\star = 0$ gives a constant $C^\star$ of the order of $(1/\rho)^d$.

Besides, in Example 3.3, we can check that $d^\star(L_0) = d/2$, so that the near-optimality dimension corresponds here to the worst-case value of $d^\star$ among the linear and quadratic regimes. Our bounds in Sections 3.3 and 3.4 do not depend on this worst-case value but instead combine all best values of $d^\star$ at all scales ε.

3.5.3 Proofs of Corollaries 3.1 and 3.3

In this section we prove Corollaries 3.1 and 3.3 stated in Section 3.3. We begin by Corollary 3.1, which gives a sample complexity bound for the Piyavskii–Shubert algorithm in terms of the near-optimality dimension of f.

Proof of Corollary 3.1. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6). Let also $\varepsilon \in (0,\varepsilon_0)$ and $\alpha \in [0,\varepsilon/25]$. Then

$$\begin{aligned}
\widetilde{n} &= \sum_{s=1}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil} \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-s}, \varepsilon_0 2^{-s+1}]}, \frac{\varepsilon_0 2^{-s} - 3\alpha}{L_1}\right) + 1 && \text{(by (3.9))}\\
&\leqslant \sum_{s=1}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil} \mathcal{N}\left(\mathcal{X}_{\varepsilon_0 2^{-s+1}}, \frac{\varepsilon_0 2^{-s} - 3\alpha}{L_1}\right) + 1 \\
&\leqslant \sum_{s=1}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil} \left(1 + 4\frac{\frac{\varepsilon_0 2^{-s}}{L_0}}{\frac{\varepsilon_0 2^{-s} - 3\alpha}{L_1}}\right)^d \mathcal{N}\left(\mathcal{X}_{\varepsilon_0 2^{-s+1}}, \frac{\varepsilon_0 2^{-s}}{L_0}\right) + 1 && \text{(by Lemma 3.6)}\\
&\leqslant \left(1 + 4\frac{L_1}{L_0}\frac{\varepsilon}{\varepsilon - 6\alpha}\right)^d \left(\sum_{s=1}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil} \mathcal{N}\left(\mathcal{X}_{\varepsilon_0 2^{-s+1}}, \frac{\varepsilon_0 2^{-s+1}}{2L_0}\right)\right) + 1 && (x \mapsto \tfrac{x}{x-3\alpha} \text{ is decreasing})\\
&\leqslant C^\star \left(1 + 4\frac{L_1}{L_0}\frac{\varepsilon}{\varepsilon - 6\alpha}\right)^d \left(\sum_{s=0}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon} \rceil - 1} \left(2^{d^\star}\right)^s\right) + 1 && \text{(by 3.6)}\\
&\leqslant 1 + C^\star \left(1 + 4\frac{L_1}{L_0}\frac{\varepsilon}{\varepsilon - 6\alpha}\right)^d \times
\begin{cases}
\log_2\left(\dfrac{\varepsilon_0}{\varepsilon}\right) + 1 & \text{if } d^\star = 0\\[2ex]
\dfrac{2^{d^\star}\left(\dfrac{\varepsilon_0}{\varepsilon}\right)^{d^\star} - 1}{2^{d^\star} - 1} & \text{if } d^\star > 0
\end{cases}
\end{aligned}$$

The result then follows by letting $\varepsilon \leftarrow \frac{25}{27}\varepsilon$, $\alpha \leftarrow \frac{1}{27}\varepsilon$, and applying Theorem 3.1. $\square$

We now show the analogous proof for automatic stopping, which concludes this section.

Proof of Corollary 3.3. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6), $\varepsilon \in (0,\varepsilon_0)$, and $\alpha \in [0,\varepsilon/24)$. Then

$$\begin{aligned}
\widetilde{n}' &\leqslant \mathcal{N}\left(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon-\alpha}{L_1}\right) + \sum_{k=0}^{\lceil \log_2 \frac{\varepsilon_0}{\varepsilon}\rceil} \mathcal{N}\left(\mathcal{X}_{(\varepsilon_0 2^{-k-1},\varepsilon_0 2^{-k}]}, \frac{\varepsilon_0 2^{-k-1}-3\alpha}{L_1}\right) && \text{(by Theorem 3.2)}\\
&\leqslant \mathcal{N}\left(\mathcal{X}_{\varepsilon/2}, \frac{\varepsilon-\alpha}{L_1}\right) + \mathcal{N}\left(\mathcal{X}_{\varepsilon}, \frac{\varepsilon/4-3\alpha}{L_1}\right) + \overline{n} && \text{(by proof of Corollary 3.1 and (3.9))}\\
&\leqslant 2\left(1+8\frac{L_1}{L_0}\frac{\varepsilon}{\varepsilon-12\alpha}\right)^d \mathcal{N}\left(\mathcal{X}_{\varepsilon}, \frac{\varepsilon}{2L_0}\right) + \overline{n} && \text{(by Lemma 3.6)}\\
&\leqslant 2\left(1+8\frac{L_1}{L_0}\frac{\varepsilon}{\varepsilon-12\alpha}\right)^d C^\star\left(\frac{\varepsilon_0}{\varepsilon}\right)^{d^\star} + \overline{n} && \text{(by (3.6))}\\
&\leqslant 1 + C^\star\left(1+8\frac{L_1}{L_0}\frac{\varepsilon}{\varepsilon-12\alpha}\right)^d \times \begin{cases} \log_2\left(\frac{\varepsilon_0}{\varepsilon}\right)+3 & \text{if } d^\star = 0\\ \dfrac{2^{d^\star}\left(\frac{\varepsilon_0}{\varepsilon}\right)^{d^\star}-1}{2^{d^\star}-1} + 2\left(\frac{\varepsilon_0}{\varepsilon}\right)^{d^\star} & \text{if } d^\star > 0 \end{cases} && \text{(by (3.10))}
\end{aligned}$$

which concludes the proof. □

3.5.4 Connections with a known regret bound in dimension one

Hansen et al. [40] provided an extensive study of the ε-Piyavskii–Shubert algorithm without perturbations (Algorithm 11 with $\alpha = 0$) for Lipshitz functions defined on a compact interval. Theorems 4 and 5 in [40] imply the following upper bound on the number n_{Py} of iterations performed by the ε-Piyavskii–Shubert algorithm before stopping.

Theorem 3.4 (Hansen et al. [40]). *Assume that $D = [0,1]$, f is globally L_0-Lipschitz on $[0,1]$, and let $L_1 \geqslant L_0$. Let also $\varepsilon > 0$, $\alpha = 0$, $x_1 \in D$, and assume that the ε-Piyavskii–Shubert algorithm (Algorithm 11) is run with inputs $L_1, \varepsilon, 0, x_1$. Then, the ε-Piyavskii–Shubert algorithm stops after at most n_{Py} iterations, where*

$$n_{\mathrm{Py}} := 1 + \frac{2L_0}{\ln(1+L_0/L_1)} \int_0^1 \frac{\mathrm{d}x}{f^\star - f(x) + \varepsilon}\,. \tag{3.19}$$

Using (3.6), we can further upper bound n_{Py} in terms of the pair $(d^\star, C^\star)$.

Corollary 3.5. *Assume that $D = [0,1]$, f is globally L_0-Lipschitz on $[0,1]$, and let $L_1 \geqslant L_0$. Let also $\varepsilon > 0$, $\alpha = 0$, $x_1 \in D$, and assume that the ε-Piyavskii–Shubert algorithm (Algorithm 11) is run with inputs $L_1, \varepsilon, 0, x_1$. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6). Then, the ε-Piyavskii–Shubert algorithm stops after at most $\overline{n}_{\mathrm{Py}}$ iterations, where*

$$\overline{n}_{\mathrm{Py}} := 1 + \frac{2v_1 C^\star}{\ln(1+L_0/L_1)} \times \begin{cases} \dfrac{v_1 C^\star}{2L_0}\left(2\log_2\left(\frac{\varepsilon_0}{\varepsilon}\right)+3\right) & \text{if } d^\star = 0\\ \dfrac{v_1 C^\star}{2L_0}\left(\dfrac{2^{d^\star+1}}{2^{d^\star}-1}+1\right)\left(\frac{\varepsilon_0}{\varepsilon}\right)^{d^\star} & \text{if } d^\star > 0 \end{cases}$$

Proof. This proof relies on a peeling technique similar to that of Theorem 3.1 and 3.2. Fix any $C^\star \geqslant 0$ and $d^\star \in [0,d]$ satisfying (3.6), and without loss of generality let $\varepsilon \in (0,\varepsilon_0)$. Defining again $m_\varepsilon := \lceil \log_2(\varepsilon_0 \varepsilon^{-1})\rceil \geqslant 1$, we have $[0,\varepsilon_0] = [0,\varepsilon] \cup (\varepsilon_0 2^{-m_\varepsilon}, \varepsilon_0 2^{-m_\varepsilon+1}] \cup (\varepsilon_0 2^{-m_\varepsilon+1}, \varepsilon_0 2^{-m_\varepsilon+2}] \cup (\varepsilon_0/2, \varepsilon_0]$ which in turn yields

$$\begin{aligned}
\int_0^1 \frac{\mathrm{d}x}{f^\star - f(x)+\varepsilon} &\leqslant \sum_{i=1}^{m_\varepsilon} \int_{\mathcal{X}_{(\varepsilon_0 2^{-i},\varepsilon_0 2^{-i+1}]}} \frac{\mathrm{d}x}{f^\star - f(x)+\varepsilon} + \int_{\mathcal{X}_\varepsilon} \frac{\mathrm{d}x}{f^\star - f(x)+\varepsilon}\\
&\leqslant \sum_{i=1}^{m_\varepsilon} \int_{\mathcal{X}_{(\varepsilon_0 2^{-i},\varepsilon_0 2^{-i+1}]}} \frac{\mathrm{d}x}{\varepsilon_0 2^{-i}} + \int_{\mathcal{X}_\varepsilon} \frac{\mathrm{d}x}{\varepsilon}\,.
\end{aligned}$$

Let v_1 be the volume $\int_{\mathbb{R}} \mathbb{I}_{\|x\| \leqslant 1} \, \mathrm{d}x$ of the real unit ball with respect to $\| \cdot \|$. For any a nonempty subset A of $[0,1]$, radius $\alpha > 0$, and collection $\mathcal{B}$ of $\| \cdot \|$-balls of radius α covering A, we have

$$\int_A \mathrm{d}x \leqslant \sum_{B \in \mathcal{B}} \int_B \mathrm{d}x \leqslant \alpha v_1 \cdot |\mathcal{B}| \; .$$

Plugging this in the previous inequality gives

$$\int_0^1 \frac{\mathrm{d}x}{f^\star - f(x) + \varepsilon} \leqslant \sum_{i=1}^{m_\varepsilon} \frac{1}{2^{-i}\varepsilon_0} \frac{2^{-i}\varepsilon_0}{L_0} v_1 \cdot \mathcal{M}\left(\mathcal{X}_{\left(\frac{\varepsilon_0}{2^i}, \frac{\varepsilon_0}{2^{i-1}}\right]}, \frac{2^{-i}\varepsilon_0}{L_0} \right) + \frac{1}{\varepsilon} \frac{\varepsilon}{2L_0} v_1 \cdot \mathcal{M}\left(\mathcal{X}_\varepsilon, \frac{\varepsilon}{2L_0} \right) \; ,$$

where $\mathcal{M}(A, \delta)$ denotes the minimum number of δ-balls required to cover A. Recalling (3.17) and noting $\mathcal{X}_{(\varepsilon_0 2^{-i}, \varepsilon_0 2^{-i+1}]} \leqslant \mathcal{X}_{\varepsilon_0 2^{-i+1}}$, we obtain

$$\int_0^1 \frac{\mathrm{d}x}{f^\star - f(x) + \varepsilon} \leqslant \frac{v_1}{L_0} \sum_{i=1}^{m_\varepsilon} C^\star \left(\frac{\varepsilon_0}{2^{-i+1}\varepsilon_0} \right)^{d^\star} + \frac{v_1}{2L_0} C^\star \left(\frac{\varepsilon_0}{\varepsilon} \right)^{d^\star} \leqslant \frac{v_1 C^\star}{L_0} \sum_{i=1}^{m_\varepsilon} 2^{(i-1)d^\star} + \frac{v_1 C^\star}{2L_0} \left(\frac{\varepsilon_0}{\varepsilon} \right)^{d^\star}$$

$$\leqslant \begin{cases} \dfrac{v_1 C^\star}{2L_0}(2m_\varepsilon + 1) & \text{if } d^\star = 0 \\ \dfrac{v_1 C^\star}{L_0} \dfrac{2^{m_\varepsilon d^\star} - 1}{2^{d^\star} - 1} + \dfrac{v_1 C^\star}{2L_0} \left(\dfrac{\varepsilon_0}{\varepsilon} \right)^{d^\star} & \text{if } d^\star > 0 \end{cases} \leqslant \begin{cases} \dfrac{v_1 C^\star}{2L_0} \left(2 \log_2 \left(\dfrac{\varepsilon_0}{\varepsilon} \right) + 3 \right) & \text{if } d^\star = 0 \\ \dfrac{v_1 C^\star}{2L_0} \left(\dfrac{2^{d^\star + 1}}{2^{d^\star} - 1} + 1 \right) \left(\dfrac{\varepsilon_0}{\varepsilon} \right)^{d^\star} & \text{if } d^\star > 0 \end{cases}$$

which gives the result. □

Chapter 4

Repeated A/B testing

In this chapter we study a setting in which a learner faces a sequence of A/B tests and has to make as many good decisions as possible within a given amount of time. Each A/B test n is associated with an unknown (and potentially negative) reward $\mu_n \in [-1, 1]$, drawn i.i.d. from an unknown and fixed distribution. For each A/B test n, the learner sequentially draws i.i.d. samples of a $\{-1, 1\}$-valued random variable with mean μ_n until a halting criterion is met. The learner then decides to either accept the reward μ_n or to reject it and get zero instead. We measure the learner's performance as the sum of the expected rewards of the accepted μ_n divided by the total expected number of used time steps (which is different from the expected ratio between the total reward and the total number of used time steps). We design an algorithm and prove a data-dependent regret bound against any set of policies based on an arbitrary halting criterion and decision rule. Though our algorithm borrows ideas from multiarmed bandits, the two settings are significantly different and not directly comparable. In fact, the value of μ_n is never observed directly in our setting—unlike rewards in stochastic bandits. Moreover, the particular structure of our problem allows our regret bounds to be independent of the number of policies.

4.1 Introduction and related work

The problem of performing repeated randomized trials for comparing statistical hypotheses dates back to the fifties [91]. With the advent of internet companies, decision making algorithms that adhere to this paradigm witnessed a new wave of interest, and several variants of this setting have been introduced in recent years [34, 37, 43, 48, 55]. As a concrete application, consider an online advertising company that keeps on testing out new technologies in order to maximize its profit. Before deploying each new technology, the company wants to figure out if its benefits outweigh its costs. As long as a reasonable performance metric is available (e.g., time spent on a page, click-through rates, conversion of curiosity to sales), the company can perform a randomized test and make a statistically sound decision. In real life applications, however, companies are likely not interested in being absolutely sure to pick the best technology, but they want to spend as little time as possible to make these decisions because of budget constraints. Indeed, testing technologies can be expensive and might slow down the regular work flow. Therefore, discarding technologies with a small positive margin could be significantly better than investing a large amount of resources to prove that they are marginally better than the current one. The use of such data-driven sequential decisions processes is known as A/B testing [51].

These types of settings are reminiscent of multiarmed bandits [2, 18, 77], where the set of arms is the set of all decision rules (or policies) used by the decision maker to determine the outcome of an A/B test. However, the two problems are significantly different. In each round of a stochastic bandit problem, the learner picks an arm and gets to see an unbiased estimate of the expected reward of that arm. The total reward accumulated by the learner is an additive function of time, the regret is simply the difference between the total reward of the best fixed arm and the total reward of the algorithm (whose partial sums the learner gets to see after each time step), and regret bounds typically scale with the number of arms [4].

In repeated A/B testing, the learner faces a sequence of A/B tests indexed by a positive integer n. The outcome of the n-th A/B test is measured by a real number μ_n that can be either positive or negative, depending on whether the tested population B generates higher rewards than the control population A. Hence, $|\mu_n|$ represents the difference in performance between the two populations. Neither the absolute value nor the sign of μ_n can be directly observed by the learner. However, the learner can draw samples to estimate μ_n, and use this estimate to make a decision about whether to accept B or not. Given μ_n, samples are assumed to be i.i.d. with expectation μ_n. Thus drawing more samples improves the estimate of μ_n, but also makes the test run longer. This is not an issue if only one A/B test is performed. However, when a sequence of such tests is performed, one wants to maximize the number of successfully implemented new technologies in a given time window.

A typical way an A/B test is performed is the following. An increasing number of i.i.d. samples $X_{n,1}, X_{n,2}, \dots$ are gathered, and the learner uses their running average to update a confidence interval around their common expectation μ_n. As soon as this confidence interval does not contain zero, the sign of μ_n becomes known with high probability (if zero falls below the confidence interval, then $\mu_n > 0$, and vice versa). When this happens, B is accepted if and only if μ_n is positive. The major drawback of this approach is that μ_n might be arbitrarily close to zero. In this case, the number of samples needed to determine the sign of μ_n, which is of order $1/\mu_n^2$, becomes arbitrarily large. The company then spends a large amount of time on an A/B-test whose return is negligible. In hindsight, it would have been better to discard this A/B test, and hope that the next one was better, i.e., that μ_{n+1} was positive and bounded away from zero. Because of that, A/B tests are usually given a termination horizon: if after k many samples no decision can be taken, then discard this technology and move on to the next one. Denote this "early stopping" rule to compute the number of samples by τ_k and the subsequent decision by $\mathcal{D}$. A policy with termination horizon k is then a pair $(\tau_k, \mathcal{D})$. Given a set of such of policies, the goal is to learn efficiently, at the smallest cost, the optimal termination horizon $k^\star$. This number is learnable if the sequence $\{\mu_n\}_n$ has some stationarity property. For this reason we assume that $\mu_1, \mu_2, \dots$ are i.i.d. random variables.

A crucial point in sequential A/B testing is picking a metric to evaluate each learning algorithm. As mentioned before, we aim at keeping a steady flow of improvements. For this reason, the performance of the learner is measured using the ratio between the expected reward of all accepted μ_n divided by the total expected number of samples drawn throughout the process. This choice ensures that the optimal policy $(\tau_{k^\star}, \mathcal{D})$ maximizes the ratio of expectations $\mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau_k]$, where τ_k is the number of samples drawn by policy $(\tau_k, \mathcal{D})$ and $\mathcal{D}$ is the decision of either accepting or rejecting μ_1 after drawing at most k samples (as described above). The main objective of the learner is to minimize the notion of regret defined as the difference between the maximum of the ratio $\mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau_k]$ (over a possibly infinite set of policies) and the learner's performance defined above. As we discuss in depth in Section 4.3, this choice is also well-suited to model sequential A/B testing when the learners can perform as many A/B tests as they want, as long as the total number of samples does not exceed a budget T. That is, when the constraint is on the number of samples rather than on the number of A/B tests.

Another possible metric that is used to optimize sequential A/B-tests is the *false discovery rate* (FDR) —see [75, 96] and references therein. Roughly speaking, the FDR in our setting would be the ratio of implemented μ_n that are negative over the total number of implemented μ_n. This quantity is usually controlled by looking at the p-values of tests and accepting (resp., rejecting) a test if its p-value is below (resp., above) some dynamically adapted threshold. This is an interesting and fruitful theory, but unfortunately disregards the relative "performance" of tests, i.e., the values of μ_n. For instance, assume that the samples $X_{n,i}$ belong to $\{-1, 1\}$, and μ_n can only take two values: $+\varepsilon$ with probability 0.9 and $-\varepsilon$ with probability 0.1, for some $\varepsilon > 0$. Then a company could implement all A/B tests immediately after the first sample. This would provide a ratio of the values of accepted tests over the number of samples of 0.8ε. To control the FDR, one would have to sample each policy approximately $1/\varepsilon^2$ times, yielding a ratio of order of ε^3. Another simple strategy that would outperform the FDR approach is simply accepting μ_n after the first sample if and only if the sample is positive. A direct computation shows that, in this case, the ratio of the values of accepted tests over the number of samples has order 0.4ε. Our approach departs from online FDR [28, 76] by taking the relative value of A/B tests into account. It is acceptable to accept a few slightly negative A/B tests as long as they are compensated by accepting many positive ones. The optimal ratio of FDR should actually be data-dependent and, as a consequence, unknown to the learner beforehand.

It turns out that running a decision making policy during an A/B test generates a collection of samples that —in general— cannot be put together in an unbiased estimate of the policy reward. Correcting this bias is a non-trivial challenge arising from our setting, and a major difference with multiarmed bandits. Moreover, the performance measure that we use is not additive in the number of A/B tests (nor in the number of samples). Therefore, algorithms have to be analyzed differently, and bandit-like techniques — where the regret is controlled during each time step and then summed over rounds— cannot be directly applied. On the positive side, the structure of our setting can be leveraged to derive regret bounds that are independent of the number of policies, another significant difference with standard multiarmed bandits. This problem can be thought of as a multiarmed bandit setting with non-additive regret in which the (biased) per-round feedback is computed by solving a best-arm identification problem [36] over two arms. We believe these technical difficulties are a necessary price to pay in order to define a realistic setting, applicable to real-life scenarios. Indeed, an important contribution of this chapter is the definition of a novel setting with a suitable performance measure.

In section4.4, we present an algorithm whose analysis can be applied to a broad class of policies. The algorithm is divided in three consecutive phases, each performing a specific task. During the first phase, policies requesting a progressively increasing amount of samples are tested, until one of them is found to have a strictly positive reward with high probability. Leveraging the definition of our performance measure, an estimate of the reward of this policy can be used to bound from above the expected number of samples drawn by the optimal policy (or policies, if there are more than one). With this we can determine a finite set of potentially optimal policies whose cardinality is *independent* of the total number of policies (indeed, the initial set of policies can even be infinite). This is a peculiar feature of our problem that, to the best of our knowledge, has not appeared elsewhere. During phases 2 and 3, the algorithm proceeds in an explore-then-exploit fashion. In phase 2 (exploration), it performs enough A/B tests, drawing enough samples for each one of them, in order to accurately estimate the expected reward per expected sample size of all potentially optimal policies. During phase 3 (exploitation) the algorithm consistently runs the policy with the highest of such estimates. For this algorithm we prove a high-probability data-dependent regret bound of order $\widetilde{\mathcal{O}}(N^{-1/3})$, independent of the number of policies, where N is the total number of A/B tests.

4.2 Problem definition and notation

We say that a function $\tau\colon \{-1,1\}^{\mathbb{N}} \to \mathbb{N}$ is a *horizon* if $\tau(\boldsymbol{x})$ only depends on the first $\tau(\boldsymbol{x})$ variables in $\boldsymbol{x} = (x_1, x_2, \ldots)$. In the introduction, we mentioned a few possible horizons: collect at most T samples, and stop before if 0 is outside some confidence intervals. A function $\mathcal{D}\colon \mathbb{N} \times \{-1,1\}^{\mathbb{N}} \to \{0,1\}$ is a *decision (rule)* if $\mathcal{D}(k, \boldsymbol{x})$ only depends on k and the first k variables in $\boldsymbol{x}$. In the introduction, the decision rule was to implement an A/B test ($D = 1$) if 0 was below the aforementioned confidence interval. We call the pair $\pi = (\tau, \mathcal{D})$ a *policy*.
Fix a (finite or countable) set of policies Π (known to the learner), and a distribution μ on $[-1,1]$ (unknown to the learner). We study the following online protocol: for each A/B test $n = 1, 2, \ldots$

1. an unobserved sample μ_n, called *mutation*, is drawn i.i.d. according to μ,
2. the learner *runs a policy* $(\tau_n, \mathcal{D}_n)$, i.e., the learner makes a decision $\mathcal{D}_n$ after drawing $\{-1,1\}$-valued samples $X_{n,1}, \ldots, X_{n,\tau_n}$ such that $\mathbb{E}[X_{n,i} \mid \mu_n] = \mu_n$, where $\tau_n = \tau_n(\boldsymbol{X}_n)$, $\mathcal{D}_n = \mathcal{D}_n\big(\tau_n(\boldsymbol{X}_n), \boldsymbol{X}_n\big)$, and $\boldsymbol{X}_n = (X_{n,1}, X_{n,2}, \ldots)$ are drawn i.i.d. given μ_n, and independently of past A/B tests.

The goal of the learner is to minimize the regret R_N after N A/B tests, defined as

$$R_N = \sup_{(\tau,\mathcal{D})\in\Pi} \left(\frac{\mathbb{E}[\mu_1 \cdot \mathcal{D}]}{\mathbb{E}[\tau]} \right) - \frac{\sum_{n=1}^{N} \mathbb{E}[\mu_n \cdot \mathcal{D}_n]}{\sum_{m=1}^{N} \mathbb{E}[\tau_m]} \tag{4.1}$$

where $\tau = \tau(\boldsymbol{X}_1)$, $\mathcal{D} = \mathcal{D}\big(\tau(\boldsymbol{X}_1), \boldsymbol{X}_1\big)$, expectations are taken with respect to the random draw of μ_n and $\boldsymbol{X}_n$.

For each policy belonging to Π, we allow the learner to reject any mutation regardless of the outcome of the sampling, and to draw arbitrarily more samples of mutations (provided these additional samples are not taken into account in the decision). Formally, for all policies $\big(\tau, \mathcal{D}(\tau)\big) \in \Pi$ and all $k \in \mathbb{N}$, the learner has access to the policies $(\tau, 0)$ and $\big(\tau + k, \mathcal{D}(\tau)\big)$. Note that invoking the power to reject a mutation μ_n after $\tau_n(\boldsymbol{X}_n)$ samples increases the cost of sampling in Equation (4.1) by $\mathbb{E}\big[\tau_n(\boldsymbol{X}_n)\big]$ while not increasing the reward in the numerator; indeed $\mathbb{E}\Big[\mu_n \cdot \mathcal{D}_n\big(\tau_n(\boldsymbol{X}_n), \boldsymbol{X}_n\big)\Big] = \mathbb{E}[\mu_n \cdot 0] = 0$ for all $\big(\tau_n, \mathcal{D}_n(\tau_n)\big) = (\tau, 0)$, with $\big(\tau, \mathcal{D}(\tau)\big) \in \Pi$ for some decision $\mathcal{D}$. Similarly, using a policy $\big(\tau + k, \mathcal{D}(\tau)\big)$ rather than $\big(\tau, \mathcal{D}(\tau)\big) \in \Pi$ has no effect on the numerator but increases the cost in the denominator by k.

4.3 Choice of performance measure

In this section we discuss our choice of performance measure. More precisely, we compare several different benchmarks and discuss how things differ if the learner has a budget of samples rather than A/B tests. We see that all choices but one are essentially equivalent and the last one —perhaps the most natural— is surprisingly poorly suited to model our problem.

At a high level, a learner constrained by a budget would like to maximize its reward per "time step". This can be done in several different ways. If the constraint is on the number N of A/B tests, then the learner might want to maximize the objective $g_1(N)$ defined by

$$g_1(N) = \mathbb{E}\Big[\sum_{n=1}^{N} \mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big) / \sum_{m=1}^{N} \tau(\boldsymbol{X}_m)\Big].$$

This is equivalent to our choice of maximizing $\mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau]$, indeed, multiplying both the numerator and the denominator by $1/N$ and applying Hoeffding's inequality gives $g_1(N) = \Theta\big(\mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau]\big)$. Furthermore, by the law of large numbers and Lebesgue's dominated convergence theorem, $g_1(N) \to \mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau]$ when $N \to \infty$.

Assume now that the constraint is on the number of samples. We say that the learner has a *budget of samples T* if as soon as the total number of samples reaches T during A/B test N (which is now a random variable), they have to interrupt the run of the current policy, reject the current mutation μ_N, and end the process. Formally, the random variable N that counts the total number of A/B tests performed by repeatedly running policy $(\tau, \mathcal{D})$ is defined by

$$N = \min\left\{ m \in \mathbb{N} \mid \sum_{n=1}^{m} \tau(\boldsymbol{X}_n) \geqslant T \right\} \ .$$

In this case, the learner might want to maximize the objective

$$g_2(T) = \mathbb{E}\left[\frac{\sum_{n=1}^{N-1} \mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big)}{T}\right] = \frac{\mathbb{E}\left[\sum_{n=1}^{N-1} \mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big)\right]}{T},$$

where the sum stops at $N-1$ because the the last A/B test is interrupted and no reward is gained. Note first that for all deterministic functions f and all $n \in \mathbb{N}$, the random variable $f(\boldsymbol{X}_n)$ is independent of $\mathbb{I}\{n \leqslant N\}$; indeed $\mathbb{I}\{N \geqslant n\} = \mathbb{I}\big\{\sum_{i=1}^{n-1} \tau(\boldsymbol{X}_i) < T\big\}$ depends on $\tau(\boldsymbol{X}_1), \dots, \tau(\boldsymbol{X}_{n-1})$ only. Hence

$$\mathbb{E}\Big[\mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big) \cdot \mathbb{I}\{N \geqslant n\}\Big] = \mathbb{E}\Big[\mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big)\Big] \cdot \mathbb{P}(N \geqslant n) \ ,$$
$$\mathbb{E}\big[\tau(\boldsymbol{X}_n) \cdot \mathbb{I}\{N \geqslant n\}\big] = \mathbb{E}\big[\tau(\boldsymbol{X}_n)\big] \cdot \mathbb{P}(N \geqslant n) \ .$$

Moreover, assume without loss of generality that $\mathbb{E}\big[\tau(\boldsymbol{X}_1)\big] < \infty$ and note that during each A/B test at least one sample is drawn, hence $N \leqslant T$ and

$$\sum_{n=1}^{\infty} \mathbb{E}\Big[\big|\mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big)\big| \cdot \mathbb{I}\{N \geqslant n\}\Big] \leqslant \sum_{n=1}^{T} \mathbb{E}\Big[\big|\mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big)\big|\Big] \leqslant T < \infty \;,$$

$$\sum_{n=1}^{\infty} \mathbb{E}\big[\tau(\boldsymbol{X}_n) \cdot \mathbb{I}\{N \geqslant n\}\big] \leqslant \sum_{n=1}^{T} \mathbb{E}\big[\tau(\boldsymbol{X}_n) \cdot \mathbb{I}\{N \geqslant n\}\big] \leqslant T < \infty \;.$$

We can therefore apply Wald's identity[94] to deduce

$$\mathbb{E}\left[\sum_{n=1}^{N} \mu_n \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_n), \boldsymbol{X}_n\big)\right] = \mathbb{E}[N]\,\mathbb{E}[\mu_1 \cdot \mathcal{D}] \qquad \text{and} \qquad \mathbb{E}\left[\sum_{n=1}^{N} \tau(\boldsymbol{X}_n)\right] = \mathbb{E}[N]\,\mathbb{E}[\tau]$$

which, using

$$\mathbb{E}\left[\sum_{n=1}^{N} \tau(\boldsymbol{X}_n)\right] - \mathbb{E}\big[\tau(\boldsymbol{X}_N)\big] \leqslant \mathbb{E}[T] = T = \mathbb{E}[T] \leqslant \mathbb{E}\left[\sum_{n=1}^{N} \tau(\boldsymbol{X}_n)\right]$$

and $-1 \leqslant \mu_N \cdot \mathcal{D}\big(\tau(\boldsymbol{X}_N), \boldsymbol{X}_N\big) \leqslant 1$, yields

$$\frac{\mathbb{E}[N]\mathbb{E}[\mu_1 \cdot \mathcal{D}] - 1}{\mathbb{E}[N]\mathbb{E}[\tau]} \leqslant g_2(T) \leqslant \frac{\mathbb{E}[N]\mathbb{E}[\mu_1 \cdot \mathcal{D}] + 1}{\mathbb{E}[N]\mathbb{E}[\tau] - \mathbb{E}\big[\tau(\boldsymbol{X}_N)\big]}$$

i.e., $g_2(T) = \Theta\big(\mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau]\big)$ and noting that $\mathbb{E}[N] \to \infty$ if $T \to \infty$, we have that $g_2(T) \to \mathbb{E}[\mu_1 \cdot \mathcal{D}]/\mathbb{E}[\tau]$ when $T \to \infty$.

This proves that having a budget of A/B tests, samples, or using any the three natural objectives introduced so far is essentially the same. For this reason, and to make the presentation clearer we chose to put a constraint on the number of A/B tests and maximize the ratio of expectations.

Before proceeding to design and analyze an algorithm to minimize the regret Equation (4.1), we discuss a very natural definition of objective which should be avoided because, albeit easier to maximize, it is not well-suited to model our problem. Consider as objective the average payoff of accepted mutations per amount of time used to make the decision, i.e.,

$$g_3 = \mathbb{E}\left[\frac{\mu_1 \cdot \mathcal{D}}{\tau}\right].$$

We now give some intuition on the differences between the ratio of expectations and the expectation of the ratio g_3 and we discuss why the former might be better than the latter.

Fix $c \in (0,1)$, $n \updownarrow N \in \mathbb{N}$, and assume that μ_1 is uniformly distributed over $\{-c, 0, c\}$. Consider the policy $(\tau, \mathcal{D})$ satisfying

$$\Big(\tau(\boldsymbol{X}_1), \mathcal{D}\big(\tau(\boldsymbol{X}_1), \boldsymbol{X}_1\big)\Big) = \begin{cases} (n,1), & \text{if } \mu_1 = c, \\ (n,0), & \text{if } \mu_1 = -c, \\ (N,0), & \text{if } \mu_1 = 0, \end{cases}$$

i.e., the learner understands quickly ($\tau = n$ samples) that $\mu_1 = \pm c$, accepting it or rejecting it accordingly, but takes a long time ($\tau = N \gg n$ samples) to figure out that the mutation is nonpositive when $\mu_1 = 0$. In this instance, our definition of ratio of expectations and the expectation of the ratio give

$$g_3 = \mathbb{E}\left[\frac{\mu_1 \cdot \mathcal{D}}{\tau}\right] = \Theta\left(\frac{c}{n}\right) \qquad \gg \qquad \Theta\left(\frac{c}{N}\right) = \frac{\mathbb{E}[\mu_1 \cdot \mathcal{D}]}{\mathbb{E}[\tau]}.$$

This is due to the fact that the expectation of the ratio "ignores" outcomes with null (or very small) rewards, even if a large number of samples is needed to learn them. On the other hand, the ratio of expectations weighs the number of samples and it is highly influenced by it, a property we are interested in capturing with our model.

4.4 Explore then exploit

As described in the introduction, horizons in A/B testing are usually defined by a capped early-stopping rule (e.g., drawing samples until 0 leaves a confidence interval around the empirical average, but quitting with a rejection after a certain number of samples has been drawn). In this section we design an algorithm that achieves vanishing regret (with high probability) for an infinite family of policies depending on an arbitrary decision and any monotone sequence of capped horizons, not necessarily defined as truncated versions of a base early-stopping rule. Our algorithm falls in the category of the explore then exploit algorithms[71]. There are two separate phases of the algorithm dedicated to the estimation of the performance of all promising policies (exploration) and the consistent run of the policy with the best estimated performance (exploitation).

Fix any decision $\mathcal{D}$ and any sequence of horizons $\tau^1, \tau^2, \ldots$ such that $\tau^k \leqslant k$ for all k and $\tau^k \leqslant \tau^h$ if $k \leqslant h$. Consider the set of policies (parameterized by $\mathcal{D}, \tau^1, \tau^2, \ldots$)

$$\Pi = \{\pi_k\}_{k\in\mathbb{N}} = \Big\{\big(\tau^k, \mathcal{D}\big)\Big\}_{k\in\mathbb{N}} \ . \tag{4.2}$$

We say that k is the *maximum horizon* of policy $\big(\tau^k, \mathcal{D}\big)$. We denote the (expected) reward of policy $\big(\tau^k, \mathcal{D}\big)$ by

$$r_k = \mathbb{E}\Big[\mu_1 \cdot \mathcal{D}\big(\tau^k(\boldsymbol{X}_1), \boldsymbol{X}_1\big)\Big]$$

and its (expected) horizon by

$$s_k = \mathbb{E}\big[\tau^k(\boldsymbol{X}_1)\big] \ .$$

We say that $k^\star$ is an *optimal (maximum) horizon* if it satisfies

$$k^\star \in \operatorname*{arg\,max}_{k\in\mathbb{N}} (r_k/s_k) \ .$$

Namely, if it is the maximum horizon of an optimal policy $\big(\tau^{k^\star}, \mathcal{D}\big)$. Note that $\tau^k(\boldsymbol{x}) \leqslant \tau^h(\boldsymbol{x})$ for all $k \leqslant h$ and $\boldsymbol{x} \in \{-1,1\}^{\mathbb{N}}$ implies

$$s_k = \mathbb{E}\big[\tau^k(\boldsymbol{X}_1)\big] \leqslant \mathbb{E}\big[\tau^h(\boldsymbol{X}_1)\big] = s_h \ . \tag{4.3}$$

If $k \geqslant 1$, $h \geqslant 0$, and policy $(k+h, 0)$ is run during A/B test n, we denote the empirical average of the first k and the last h samples of μ_n by

$$\overline{X}^n_k = \frac{1}{k}\sum_{i=1}^{k} X_{n,i} \qquad \text{and} \qquad \overline{Y}^n_h = \frac{1}{h}\sum_{i=k+1}^{k+h} X_{n,i}$$

respectively, where $\overline{Y}^n_0$ is defined as 0. If policy $(k+h, 0)$ is run for n_2 consecutive A/B tests $n_1+1, n_1+2, \ldots, n_1+n_2$, for all $\varepsilon > 0$ we define the upper confidence bound $\widehat{r}^{\,+}_{k,h}(\varepsilon)$ of r_k and the lower confidence bounds $\widehat{r}^{\,-}_{k,h}(\varepsilon)$ of r_k and $\widehat{s}^{\,-}_{k,h}(\varepsilon)$ of s_k by

$$\widehat{r}^{\,\pm}_{k,h}(n_1, n_2, \varepsilon) = \frac{1}{n_2}\sum_{n=n_1+1}^{n_1+n_2} \overline{Y}^n_h \cdot \mathcal{D}\big(\tau^k(\boldsymbol{X}_n), \boldsymbol{X}_n\big) \pm \varepsilon \qquad \text{and} \qquad \widehat{s}^{\,-}_k(n_1, n_2, \varepsilon) = \frac{1}{n_2}\sum_{n=n_1+1}^{n_1+n_2} \tau^k(\boldsymbol{X}_n) - \varepsilon \ .$$

If policy $\big(\tau^k, 0\big)$ is run for m_0 consecutive A/B tests $n_0+1, n_0+2, \ldots, n_0+m_0$, we denote the empirical average of its horizon by

$$\overline{s}_k(n_0, m_0) = \frac{1}{m_0}\sum_{i=n_0+1}^{n_0+m_0} \tau^k(\boldsymbol{X}_i) \ .$$

For all $\varepsilon, \delta \in (0,1)$ and all $j \in \mathbb{N}$, let

$$m_j(\varepsilon, \delta) = \left\lceil \frac{\ln\big(3j(j+1)/\delta\big)}{(2\varepsilon^2)} \right\rceil \qquad \text{and} \qquad M_j = \sum_{i=1}^{j} m_i \ .$$

```
Algorithm 13:
  Input: decision $\mathcal{D}$, horizons $\tau^1, \tau^2, \dots$, budget of mutations $N \in \mathbb{N}$, confidence $1-\delta \in (0,1)$,
         accuracy $\varepsilon \in (0,1)$, exploration length $N_2 \in \mathbb{N}$, number of extra samples $h_1, h_2 \in \mathbb{N}$.
  Initialization: let $m_j \leftarrow m_j(\varepsilon,\delta)$ for all $j \in \mathbb{N}$.
1  for $j = 1, 2, \dots$ do                                   // Phase 1: Upper bounding optimal horizons
2      run policy $\big(2^j + h_1, 0\big)$ for $m_j$ A/B tests and compute $\widehat{r}^{\,-}_{2^j} \leftarrow \widehat{r}^{\,-}_{2^j,h_1}(M_{j-1}, m_j, \varepsilon)$ if $\widehat{r}^{\,-}_{2^j} > 0$ then
3          l
4      et $j_0 \leftarrow j$ and $k_0 \leftarrow 2^{j_0}$  for $l = j_0+1, j_0+2, \dots$ do
5          run policy $\big(\tau^{(2^l)}, 0\big)$ for $m_l$ A/B tests and compute $\overline{s}_{2^l} \leftarrow \overline{s}_{2^l}(M_{l-1}, m_l)$ if $\overline{s}_{2^l} > \varepsilon + k_0/\widehat{r}^{\,-}_{k_0}$
             then
6              let $j_1 \leftarrow l$, $k_2 \leftarrow 2^{j_1}$, $N_1 \leftarrow \sum_{j=1}^{j_1} m_j$, and break
7      break
8  for A/B test $n = N_1 + 1$ to $N_1 + N_2$ do                                  // Phase 2: Exploration
9      run policy $(k_2 + h_2, 0)$
10 for A/B test $n = N_1 + N_2 + 1$ to $N$ do                                     // Phase 3: Exploitation
11     run policy $\pi_{k'}$, where $k' \in \arg\max_{k \in \{1,\dots,k_2\}} \big(\widehat{r}^{\,+}_k / \widehat{s}^{\,-}_k\big)$ and

           $\widehat{r}^{\,+}_k \leftarrow \widehat{r}^{\,+}_{k,h_2}\big(N_1, N_2, \ln(6k_2\delta^{-1})/(2N_2)\big)$,  $\widehat{s}^{\,-}_k \leftarrow \widehat{s}^{\,-}_{k,h_2}\big(N_1, N_2, \ln(6k_2\delta^{-1})/(2N_2)\big)$
```

Algorithm 13 is divided into three phases. During phase 1 (lines 1–7), N_1 mutations (all rejected by the algorithm) are used to determine a horizon k_2 that upper bounds all optimal horizons $k^\star$ (there could be multiple optimal horizons) with high probability. This is possible because of the structure of our problem. Indeed, each optimal policy $\pi_{k^\star}$ satisfies $r_k/s_k \leqslant r_{k^\star}/s_{k^\star} \leqslant 1/s_{k^\star}$ which implies that for all policies π_k with $r_k > 0$, the expected number of samples drawn by $\pi_{k^\star}$ satisfies $s_{k^\star} \leqslant s_k/r_k$. With this in mind, we first upper bound with high probability the expected number of samples $s_{k^\star}$ drawn by optimal policies $\pi_{k^\star}$ using an estimate of some s_k/r_k with $r_k > 0$ (lines 1–2). Then we proceed to finding the smallest (up to a factor of 2) k_2 such that $s_{k_2} \geqslant s_{k^\star}$ with high probability (lines 2–7). This is an upper bound on all optimal horizons $\tau^{k^\star}$. During phase 2 (lines 8–9), the algorithm draws $k_2 + h_2$ samples from each one of the next N_2 mutations (again, all rejected). These are used to compute accurate estimates of all rewards of potentially optimal policies, i.e., all policies with maximum horizon at most k_2. During phase 3 (10–11), the algorithm runs on the remaining $N - N_1 - N_2$ mutations the policy $\big(\tau^{k'}, \mathcal{D}\big)$ that it estimated to be optimal by the end of phase 2.

Theorem 4.1. *Let Π be defined by Equation (4.2). If Algorithm 13 is run with decision $\mathcal{D}$, horizons $\tau^1, \tau^2, \dots$, budget of mutations $N \in \mathbb{N}$, confidence $1-\delta \in (0,1)$, accuracy $\varepsilon \in (0,1)$, exploration length $N_2 \in \mathbb{N}$, number of extra samples $h_1, h_2 \in \mathbb{N}$, and $8\ln(6k_2\delta^{-1}) < N_2 < N - N_1$, then, with probability at least $1-\delta$, its regret satisfies*

$$R_N = \mathcal{O}\left(\frac{N_2 + 1/\varepsilon^2}{N - N_2} + \sqrt{\frac{\ln(k_2/\delta)}{N_2}} \right) .$$

In particular, picking $h_1 = h_2 = 1$, $\varepsilon = \Theta(N^{-1/3})$, and $N_2 = \Theta\big(N^{2/3}\big)$ gives $R_N = \widetilde{\mathcal{O}}\big(N^{-1/3}\big)$ with probability at least $1-\delta$. Note that in general one cannot pick a constant value of ε (such as, say, $\varepsilon = 1/2$). Indeed k_2 depend on ε and the theorem holds only if phase 1 is terminated successfully initializing k_2 at line 5. This is stated (with a slight abuse of notation) in the condition $8\ln(6k_2\delta^{-1}) < N - N_1$, which implicitly states that the number N_1 of A/B tests performed during phase 1 has to be strictly smaller than the total budget of A/B tests N. Picking a large ε might result in the tests at lines 2 or 5 to never be true, so that the algorithm neither leaves phase 1, nor initializes k_2. However, if ε is set to $\Theta(N^{-1/3})$ and $N \gg 1$, a direct verification shows that this can only happen if all optimal policies $\pi_{k^\star}$ satisfy $r_{k^\star}/s_{k^\star} = \mathcal{O}(N^{-1/3})$, which in turn ensures that the regret is still of order $N^{-1/3}$. The same argument applies to the other constants that appear inside the the big O notation (see proof below for the explicit form of the bound). Note that this bound is independent of the number of policies (which would be infinite). It scales instead with the data-dependent constant k_2, i.e., the upper bound on all optimal horizons, which has a clear interpretation as a measure of hardness of the instance. The more μ_n is concentrated around 0, the hardest the problem of determining if $\mu_n > 0$ becomes and the more samples are needed in order to prove it.

Proof. Note that N_1, N_2, and $N - N_1 - N_2 \geqslant 1$ are the numbers of A/B tests performed during phases 1, 2, and 3 respectively. The total number of samples drawn during phase 1 is

$$\sum_{j=1}^{j_0}(2^j + h_1)m_j + \sum_{l=j_0+1}^{j_1} 2^j m_j = h_1 M_{j_0} + \sum_{j=1}^{j_1} 2^j m_j$$
$$\leqslant h_1 M_{j_0} + \sum_{j=1}^{j_1} 2^j m_{j_1} = h_1 M_{j_0} + 2\left(\frac{1-2^{j_1}}{1-2}\right) m_{j_1} = h_1 M_{j_0} + (2k_1 - 2)m_{j_1} \ .$$

Since all mutations are rejected, the sum of the rewards accumulated during phase 1 is zero. The total number of samples drawn during phase 2 is equal to $(k_2 + h_2)N_2$. Since all mutations are rejected, the sum of the rewards accumulated during phase 2 is again zero. The total expected number of samples drawn during phase 3 is upper bounded by $(N - N_2)s_{k'}$. We now proceed to lower bound the total amount of reward $(N - N_1 - N_2)r_{k'}$ accumulated during phase 3. We begin by showing that $k^\star \leqslant k_2$ for all optimal horizons $k^\star$ with high probability. Note that $\widehat{r}^{\,-}_{2^j} + \varepsilon$ (line 2) is an empirical average of i.i.d. unbiased estimators of r_{2^j}. Indeed, by the independence of $\mathcal{D}(k, \boldsymbol{x})$ of the variables $(x_{k+1}, x_{k+2}, \dots)$ and the conditional independence of the samples, for all A/B tests n performed at line 2,

$$\mathbb{E}\left[\overline{Y}^n_{h_1} \cdot \mathcal{D}\big(\tau^{(2^j)}(\boldsymbol{X}_n), \boldsymbol{X}_n\big) \mid \mu_n\right] = \mathbb{E}\left[\overline{Y}^n_{h_1} \mid \mu_n\right] \mathbb{E}\Big[\mathcal{D}\big(\tau^{(2^j)}(\boldsymbol{X}_n), \boldsymbol{X}_n\big) \mid \mu_n\Big]$$
$$= \mu_n \cdot \mathbb{E}\Big[\mathcal{D}\big(\tau^{(2^j)}(\boldsymbol{X}_n), \boldsymbol{X}_n\big) \mid \mu_n\Big] = \mathbb{E}\Big[\mu_n \cdot \mathcal{D}\big(\tau^{(2^j)}(\boldsymbol{X}_n), \boldsymbol{X}_n\big) \mid \mu_n\Big] \ .$$

Taking expectations to both sides proves the claim. Then, recalling that $M_j = \sum_{i=1}^j m_i$ and denoting by Σ_j the σ-algebra generated by $\mu_{M_{j-1}+1}, \dots, \mu_{M_j}$, Hoeffding's inequality implies

$$\mathbb{P}\left(\widehat{r}^{\,-}_{2^j,h_1}(M_{j-1}, m_j, \varepsilon) > r_{2^j} \mid \Sigma_j\right) = \mathbb{P}\Big(\big(\widehat{r}^{\,-}_{2^j,h_1}(M_{j-1}, m_j, \varepsilon) + \varepsilon\big) - r_{2^j} > \varepsilon \mid \Sigma_j\Big) \leqslant \frac{\delta}{3j(j+1)}$$

for all $j \leqslant j_0$. By the law of total expectation, the same holds without the conditioning. Similarly, for all $l > j_0$, $\mathbb{P}(\overline{s}_{2^j} - s_{2^l} > \varepsilon) \leqslant \delta/\big(3l(l+1)\big)$. Hence, the event

$$\left\{\widehat{r}^{\,-}_{2^j} \leqslant r_{2^j}\right\} \wedge \left\{\overline{s}_{2^j} \leqslant s_{2^j} + \varepsilon\right\} \qquad \forall j \leqslant j_0, \forall l > j_0 \tag{4.4}$$

occurs with probability at least

$$1 - \sum_{j=1}^{j_0} \frac{\delta}{3j(j+1)} - \sum_{l=j_0+1}^{j_1} \frac{\delta}{3l(l+1)} \geqslant 1 - \frac{\delta}{3} \sum_{j\in\mathbb{N}} \frac{1}{j(j+1)} = 1 - \frac{\delta}{3} \ .$$

Note now, that for all horizons k and all optimal horizons $k^\star$,

$$\frac{r_k}{k} \leqslant \frac{r_k}{s_k} \leqslant \frac{r_{k^\star}}{s_{k^\star}} \leqslant \frac{1}{s_{k^\star}} \ .$$

Hence, all optimal horizons $k^\star$ satisfy $s_{k^\star} \leqslant k/r_k$ for all horizons k such that $r_k > 0$. By Equation (4.3), Equation (4.4) and line 5, with probability at least $1 - \delta/3$, for all $k > k_2$, $s_k \geqslant s_{k_2} \geqslant \overline{s}_{k_2} - \varepsilon > k_0/\widehat{r}^{\,-}_{k_0} \geqslant$

k_0/r_{k_0}, where $r_{k_0} \geqslant \widehat{r}_{k_0}^- > 0$ by Equation (4.3) and line Equation (2). Therefore, all optimal horizons $k^\star$ satisfy $k^\star \leqslant k_2$ with probability at least $1 - \delta/3$.

Now we show that the policy $\big(k', \mathcal{D}(k')\big)$ determined by the end of phase 2 is a close approximation of all optimal policies $\big(k^\star, \mathcal{D}(k^\star)\big)$ with high probability. Proceeding as above yields, with probability at least $1 - 2\delta/3$, for all $k \in \{1, \dots, k_2\}$, $r_k \leqslant \widehat{r}_k^+ \leqslant r_k + 2\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}$ and $s_k \geqslant \widehat{s}_k^- \geqslant s_k - 2\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}$. Then, using $(a+b)/(c-b) \leqslant a/b + 2b/(c-b)$ for all $a, b, c > 0$ with $c > b$ and $a/b \leqslant 1$, we have with probability at least $1 - \delta$, for all optimal horizons $k^\star$,

$$\frac{r_{k^\star}}{s_{k^\star}} \leqslant \frac{\widehat{r}_{k^\star}^+}{\widehat{s}_{k^\star}^-} \leqslant \frac{\widehat{r}_{k'}^+}{\widehat{s}_{k'}^-} \leqslant \frac{r_{k'} + 2\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}}{s_{k'} - 2\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}} \leqslant \frac{r_{k'}}{s_{k'}} + \frac{4\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}}{s_{k'} - 2\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}} \ .$$

That is, for all optimal horizons $k^\star$, the policy $\pi_{k'}$ played during the exploitation phase satisfies, with probability at least $1 - \delta$,

$$r_{k'} \geqslant s_{k'} \left(\frac{r_{k^\star}}{k^\star} - \frac{4\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}}{s_{k'} - 2\sqrt{\ln(6k_2\delta^{-1})/(2N_2)}} \right) > s_{k'} \left(\frac{r_{k^\star}}{k^\star} - \sqrt{\frac{32\ln(6k_2\delta^{-1})}{N_2}} \right)$$

where we used $N_2 > 8\ln(6k_2\delta^{-1})$ in the last inequality. Putting everything together and using $\frac{a}{a+b}(x-y) \geqslant x - y - \frac{b}{b+c}$ for all $a, b, y > 0$ and all $x \leqslant 1$ gives, with probability at least $1 - \delta$, for all optimal horizons $k^\star$,

$$R_T \leqslant \frac{h_1 M_{j_0} + (2k_1 - 2)m_{j_1} + (k_2 + h_2)N_2}{h_1 M_{j_0} + (2k_1 - 2)m_{j_1} + (k_2 + h_2)N_2 + (N - N_2)s_{k'}} + \sqrt{\frac{32\ln(6k_2/\delta)}{N_2}} \ .$$

□

4.5 Conclusions and future work

In this chapter we introduced a sequential A/B testing problem in which the goal of the learner is to simultaneously maximize the reward accumulated by accepting mutations and minimize the number of samples necessary to do so. While we managed to design a general algorithm with vanishing regret, some interesting questions remain open.

The explore-then-exploit approach could be replaced with a more adaptive online protocol. This might lead to a faster convergence rate (e.g., $\sqrt{N}$). The issue that emerges from directly applying UCB [4] strategies to this problem is that those algorithms typically rely on building upper confidence bounds around empirical averages of unbiased estimates of the reward. As we discussed earlier, the estimates we get by running a policy without oversampling are biased. It is not clear whether such biased estimates could be used, as their non-vanishing bias depends in a non-trivial way on the form of the policy. One could however get unbiased estimates by drawing as little as one extra sample for each A/B test. This would ensure fast convergence, but to a slightly suboptimal policy. In fact, this algorithm would have a competitive ratio of $s_{k^\star}/(s_{k^\star}+1) > 1$, unlike a proper regret bound. Other approaches, such as explore then exploit with a more adaptive exploration, might yield better regret guarantees. These lines of research will be explored in future works.

Bibliography

[1] Shipra Agrawal and Nikhil R Devanur. Bandits with concave rewards and convex knapsacks. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 989–1006. ACM, 2014.

[2] Noga Alon, Nicolo Cesa-Bianchi, Ofer Dekel, and Tomer Koren. Online learning with feedback graphs: Beyond bandits. In *Annual Conference on Learning Theory*, volume 40. Microtome Publishing, 2015.

[3] Kareem Amin, Afshin Rostamizadeh, and Umar Syed. Learning prices for repeated auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, pages 1169–1177, 2013.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[5] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. on Computing*, 32(1):48–77, 2002.

[6] Baruch Awerbuch and Robert Kleinberg. Competitive collaborative learning. *Journal of Computer and System Sciences*, 74(8):1271–1288, 2008.

[7] Moshe Babaioff, Shaddin Dughmi, Robert Kleinberg, and Aleksandrs Slivkins. Dynamic pricing with limited supply. *ACM Transactions on Economics and Computation (TEAC)*, 3(1):4, 2015.

[8] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 207–216. IEEE, 2013.

[9] Quentin Berthet and Vianney Perchet. Fast rates for bandit optimization with upper-confidence frank-wolfe. In *Advances in Neural Information Processing Systems*, pages 2222–2231, 2017.

[10] Avrim Blum and Jason D Hartline. Near-optimal online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1156–1163. Society for Industrial and Applied Mathematics, 2005.

[11] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. *Theoretical Computer Science*, 324(2-3):137–146, 2004.

[12] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence.* OUP Oxford, 2013.

[13] Clement Bouttier. *Global optimization under uncertainty: stochastic algorithms and continuous bandits with application to aircraft performance.* PhD thesis, 2017.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[15] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical report, 2010. arXiv:1012.2599.

[16] Josef Broder and Paat Rusmevichientong. Dynamic pricing under a general parametric choice model. *Operations Research*, 60(4):965–980, 2012.

[17] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015. ISSN 1935-8237.

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

[19] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

[20] Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Bounded regret in stochastic multi-armed bandits. In *Conference on Learning Theory*, pages 122–134, 2013.

[21] Sebastien Bubeck, Nikhil R Devanur, Zhiyi Huang, and Rad Niazadeh. Online auctions and multi-scale online learning. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 497–514. ACM, 2017.

[22] Alexandra Carpentier and Michal Valko. Simple regret for infinitely many armed bandits. In *International Conference on Machine Learning*, pages 1133–1141, 2015.

[23] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games.* Cambridge University Press, 2006.

[24] Nicolo Cesa-Bianchi, Claudio Gentile, Yishay Mansour, and Alberto Minora. Delay and cooperation in nonstochastic bandits. *JMLR Workshop and Conference Proceedings (COLT 2016)*, 49:605–622, 2016.

[25] Nicolò Cesa-Bianchi, Tommaso Cesari, and Vianney Perchet. Dynamic pricing with finitely many unknown valuations. In *Algorithmic Learning Theory*, pages 247–273, 2019.

[26] Nicolò Cesa-Bianchi, Tommaso R Cesari, Yishay Mansour, and Vianney Perchet. Repeated a/b testing. *arXiv preprint arXiv:1905.11797*, 2019.

[27] Nicolò Cesa-Bianchi, Tommaso R Cesari, and Claire Monteleoni. Cooperative online learning: Keeping your neighbors updated. *arXiv preprint arXiv:1901.08082*, 2019.

[28] Shiyun Chen and Shiva Kasiviswanathan. Contextual online false discovery rate control. *arXiv preprint arXiv:1902.02885*, 2019.

[29] Arnoud den Boer and N Bora Keskin. Discontinuous demand functions: Estimation and pricing. 2017.

[30] Arnoud V den Boer. Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in operations research and management science*, 20(1):1–18, 2015.

[31] Nikhil R Devanur, Yuval Peres, and Balasubramanian Sivan. Perfect Bayesian equilibria in repeated sales. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 983–1002. SIAM, 2014.

[32] John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

[33] Rachid Ellaia, Mohamed Zeriab Es-Sadek, and Hasnaa Kasbioui. Modified Piyavskii's global one-dimensional optimization of a differentiable function. *Applied Mathematics*, 3(10):1306, 2012.

[34] Dean P Foster and Robert A Stine. α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(2):429–444, 2008.

[35] Yoav Freund, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. Using and combining predictors that specialize. In *In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*. Citeseer, 1997.

[36] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027, 2016.

[37] Christopher R Genovese, Kathryn Roeder, and Larry Wasserman. False discovery control with p-value weighting. *Biometrika*, 93(3):509–524, 2006.

[38] Jerrold R Griggs. Lower bounds on the independence number in terms of the degrees. *Journal of Combinatorial Theory, Series B*, 34(1):22–39, 1983.

[39] Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems*, pages 667–675, 2015.

[40] Pierre Hansen, Brigitte Jaumard, and S-H Lu. On the number of iterations of Piyavskii's global optimization algorithm. *Mathematics of Operations Research*, 16(2):334–350, 1991.

[41] Pierre Hansen, Brigitte Jaumard, and Shi-Hui Lu. Global optimization of univariate lipschitz functions: I. survey and properties. *Mathematical Programming*, 55(1):251–272, 1992.

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

[43] Philipp Heesen and Arnold Janssen. Dynamic adaptive multiple tests with finite sample fdr control. *Journal of Statistical Planning and Inference*, 168:38–51, 2016.

[44] Matthias Horn. Optimal algorithms for global optimization in case of unknown Lipschitz constant. *Journal of Complexity*, 22(1):50–70, 2006.

[45] Saghar Hosseini, Airlie Chapman, and Mehran Mesbahi. Online distributed optimization via dual averaging. In *52nd Annual IEEE Conference on Decision and Control (CDC)*, pages 1484–1489. IEEE, 2013.

[46] Prateek Jain and Purushottam Kar. Non-convex optimization for machine learning. *Foundations and Trends in Machine Learning*, 10(3-4):142–336, 2017.

[47] Prateek Jain, Purushottam Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–336, 2017.

[48] Adel Javanmard, Andrea Montanari, et al. Online rules for control of false discovery rate and false discovery exceedance. *The Annals of statistics*, 46(2):526–554, 2018.

[49] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[50] Richard M Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 881–890. Society for Industrial and Applied Mathematics, 2007.

[51] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of a/b testing. In *Conference on Learning Theory*, pages 461–481, 2014.

[52] Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science*, pages 594–605. IEEE, 2003.

[53] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

[54] Daniela Lera and Ya D Sergeyev. Global minimization algorithms for Hölder functions. *BIT Numerical mathematics*, 42(1):119–133, 2002.

[55] Ang Li and Rina Foygel Barber. Multiple testing with the structure-adaptive benjamini–hochberg algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(1):45–74, 2019.

[56] Cédric Malherbe and Nicolas Vayatis. Global optimization of Lipschitz functions. Technical report, 2017. arXiv: 1703.02628.

[57] Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. In *Advances in Neural Information Processing Systems*, pages 684–692, 2011.

[58] David Martínez-Rubio, Varun Kanade, and Patrick Rebeschini. Decentralized cooperative stochastic multi-armed bandits. *arXiv preprint arXiv:1810.04468*, 2018.

[59] Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample-variance penalization. In *Conference on Learning Theory*, pages 1–9, 2009.

[60] David Q Mayne and Elijah Polak. Outer approximation algorithm for nondifferentiable optimization problems. *Journal of Optimization Theory and Applications*, 42(1):19–30, 1984.

[61] R Preston McAfee. The design of advertising exchanges. *Review of Industrial Organization*, 39(3): 169–185, 2011.

[62] Scott McQuade and Claire Monteleoni. Global climate model tracking using geospatial neighborhoods. In *Proc. Twenty-Sixth AAAI Conference on Artificial Intelligence, Special Track on Computational Sustainability and AI*, pages 335–341, 2012.

[63] Scott McQuade and Claire Monteleoni. Spatiotemporal global climate model tracking. *Large-Scale Machine Learning in the Earth Sciences; Data Mining and Knowledge Discovery Series. Srivastava, A., Nemani R., Steinhaeuser, K. (Eds.), CRC Press, Taylor & Francis Group*, 2017.

[64] Regina H. Mladineo. An algorithm for finding the global maximum of a multimodal, multivariate function. *Mathematical Programming*, 34(2):188–200, 1986.

[65] Jonas Mueller, Vasilis Syrgkanis, and Matt Taddy. Low-rank bandit methods for high-dimensional dynamic pricing. *arXiv preprint arXiv:1801.10242*, 2018.

[66] Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *NIPS*, pages 783–791, 2011.

[67] Rémi Munos. From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. 2014.

[68] Dragan Nešić, Thang Nguyen, Ying Tan, and Chris Manzie. A non-gradient approach to global extremum seeking: An adaptation of the Shubert algorithm. *Automatica*, 49(3):809–815, 2013.

[69] Yurii Nesterov. *Introductory lectures on convex optimization: a basic course*, volume 87 of *Applied Optimization*. Springer US, 2004.

[70] Francesco Orabona, Koby Crammer, and Nicolo Cesa-Bianchi. A generalized online mirror descent with applications to classification and regression. *Machine Learning*, 99(3):411–435, 2015.

[71] Vianney Perchet, Philippe Rigollet, Sylvain Chassang, Erik Snowberg, et al. Batched bandit problems. *The Annals of Statistics*, 44(2):660–681, 2016.

[72] A.G. Perevozchikov. The complexity of the computation of the global extremum in a class of multi-extremum problems. *USSR Computational Mathematics and Mathematical Physics*, 30(2):28–33, 1990.

[73] S.A. Piyavskii. An algorithm for finding the absolute extremum of a function. *Comput. Math. Math. Phys.*, 12(4):57–67, 1972.

[74] Mohamed Rahal and Abdelkader Ziadi. A new extension of Piyavskii's method to Hölder functions of several variables. *Applied Mathematics and Computation*, 197(2):478–488, 2008.

[75] Aaditya Ramdas, Fanny Yang, Martin J Wainwright, and Michael I Jordan. Online control of the false discovery rate with decaying memory. In *Advances In Neural Information Processing Systems*, pages 5650–5659, 2017.

[76] David S Robertson and James Wason. Online control of the false discovery rate in biomedical research. *arXiv preprint arXiv:1809.07292*, 2018.

[77] Dinah Rosenberg, Eilon Solan, and Nicolas Vieille. Social learning in one-arm bandit problems. *Econometrica*, 75(6):1591–1611, 2007.

[78] Michael Rothschild. A two-armed bandit theory of market pricing. *Journal of Economic Theory*, 9(2): 185–202, 1974.

[79] Anit Kumar Sahu and Soummya Kar. Dist-Hedge: A partial information setting based distributed non-stochastic sequence prediction algorithm. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 528–532. IEEE, 2017.

[80] Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2745–2754, 2018.

[81] Shahin Shahrampour and Ali Jadbabaie. Distributed online optimization in dynamic environments using mirror descent. *IEEE Transactions on Automatic Control*, 63(3):714–725, 2018.

[82] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[83] Shai Shalev-Shwartz. Introduction to online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[84] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[85] Zuhe Shen and Yiran Zhu. An interval version of Shubert's iterative method for the localization of the global maximum. *Computing*, 38(3):275–280, 1987.

[86] Bruno O Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.

[87] Aleksandrs Slivkins and Assaf Zeevi. Dynamic Pricing Under Model Uncertainty. Tutorial given at the 16th ACM Conference on Economics and Computation, 2015.

[88] James C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control.* Wiley-Interscience, 2003.

[89] Francesco Trovo, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Budgeted multi–armed bandit in continuous action space. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 560–568. IOS Press, 2016.

[90] Francesco Trovò, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Improving multi-armed bandit algorithms in online pricing settings. *International Journal of Approximate Reasoning*, 98:196–235, 2018.

[91] John Wilder Tukey. The problem of multiple comparisons. *Unpublished*, 1953.

[92] Robert J Vanderbei. Extension of Piyavskii's algorithm to continuous global optimization. *Journal of Global Optimization*, 14(2):205–216, 1999.

[93] Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

[94] Abraham Wald. On cumulative sums of random variables. *The Annals of Mathematical Statistics*, 15 (3):283–296, 1944.

[95] Jonathan Weed, Vianney Perchet, and Philippe Rigollet. Online learning in repeated auctions. In *Conference on Learning Theory*, pages 1562–1583, 2016.

[96] Fanny Yang, Aaditya Ramdas, Kevin G Jamieson, and Martin J Wainwright. A framework for multi-a (rmed)/b (andit) testing with online fdr control. In *Advances in Neural Information Processing Systems*, pages 5957–5966, 2017.

[97] Yawei Zhao, Chen Yu, Peilin Zhao, and Ji Liu. Decentralized online learning: Take benefits from others' data without sharing your own to track global trend. *arXiv preprint arXiv:1901.10593*, 2019.