Online Algorithms with Learned Predictions

by

Keerti Anand

Department of Computer Science
Duke University

Date: _____

Approved:

_____
Debmalya Panigrahi, Supervisor

_____
Rong Ge

_____
Kamesh Munagala

_____
Amit Kumar

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2022

ABSTRACT

Online Algorithms with Learned Predictions

by

Keerti Anand

Department of Computer Science
Duke University

Date: _____

Approved:

_____
Debmalya Panigrahi, Supervisor

_____
Rong Ge

_____
Kamesh Munagala

_____
Amit Kumar

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University
2022

Abstract

Optimization under uncertainty is a classic theme in the fields of algorithm design and machine learning. The traditional design of online algorithms have however proved to be insufficient for practical instances, since it only tries to optimize for the worst-case (but possibly highly unlikely) future scenarios. The availability of data and the advent of powerful machine learning paradigms has led to a promising approach of leveraging predictions about the future to aid in making decisions under uncertainty. This motivates us to explore whether algorithm design can go beyond the worst-case, i.e is it possible to design efficient algorithms that perform well for the typical instances, while retaining a suitable robustness guarantee for the worst-case instance? This entails our vision of combining the power of Machine Learning and Algorithm design to get the best of both worlds. This can be categorized into two inter-dependent parts : (1) How to design the prediction pipeline to generate forecasts about the future unknowns and (2) Given such predictions, how to re-design the decision-making algorithm to best leverage them. In this thesis, we investigate both of these questions, and summarise the key contributions as follows:

Rent or Buy Problem We investigate the rent-or-buy problem and design a simple online algorithm that leverages predictions from classification black-box model whose performance is directly dependent on the prediction error of the classification task. We then demonstrate that by incorporating the optimization benchmarks in prediction model leads to significantly better performance, while maintaining a worst-case adversarial result.

Online Search We define a general online search framework that captures classic problems like (generalized) ski rental, bin packing, minimum makespan scheduling, etc. We then model the task of making predictions as a regression problem and show nearly tight bounds on the sample complexity of this regression problem.

Online Algorithms with Multiple Predictions We study the setting of online covering problems that are supplemented with multiple predictions. We give algorithmic techniques that obtain a solution that is competitive against the performance of the best predictor. We apply our algorithmic framework to solve classical problems such as online set cover, (weighted) caching, and online facility location in the multiple predictions setting.

Dedicated to the memory of my late lamented grandmother

Contents

List of Figures

List of Abbreviations and Symbols

Symbols

We put general notes about symbol usage here.

$\mathbb{D}$	Denotes a distribution.
$\mathscr{A}$	Denotes an online algorithm.
$\mathfrak{X}$	Denotes the sample space.
$\mathbb{R}^d$	Denotes d dimensional Euclidean space.
COST	Denotes the cost incurred by an algorithm.
OPT	Denotes an optimal offline algorithm.

Abbreviations

CR	Competitive Ratio.
MTS	Metric Task Systems.

Acknowledgements

This thesis would not have been possible without the mentorship and guidance of my two co-advisors, Debmalya Panigrahi and Rong Ge. They have been instrumental in helping me develop a research mindset and become able to present my ideas in a cogent and cohesive manner. They have been extremely mindful of my needs and patient with my mistakes. I would like to thank them from the bottom of my heart.

I would like to thank Amit Kumar for being my collaborator on several research projects and for very helpful discussions during my visits to IIT Delhi. I would also thank Kamesh Munagala for serving on my thesis committee.

I would not have undertaken a research initiative in Theoretical Computer Sciences had it not been for the teachings of Surender Baswana, who instilled in me an appreciation for the art of Algorithm Design when I was a newling at IIT Kanpur.

Being in Durham, N.C over the past 5 years, I have had the opportunity to get to know many people: Ashwin Shankar, Shalin Shah, Abhishek Dubey, Arjun Chaudhuri, Harshank Shrotriya, Christian Savarain, Shreya Arya, Anurag Kashyap, Amelia Moorhead, Sultan Muratov, Govind S. Shankar, Devika C. Yanamadala, Puja V.S.D. Majety, Sarisht Wadhwa. I thank them all for easing my life in a foreign land. Thanks to Kevin T. Sun and Jeffrey Ames for lending a helpful ear to my endless rants about the perils of graduate life.

Before coming to Durham, I spent the best 3.5 years of my life as an undergrad at IITK and made life-long friends whom I cherish and would like to thank, an incomplete list includes: Abhinav Soni, Vikash Chaudhary, Manish Yadav, Dasari Shanti Sree, Abhimanyu

Yadav, Mekala Dheeraj.

A special thanks goes to a very important person in my life, Sonja E.K.W. Witte who is truly the most magnificent human being I have met; her unrelenting support and unwavering faith in me have compelled me to not quit in the most trying of times.

I would like to thank my parents for bringing me into this world and ingraining me with a value system that has enabled me to work diligently towards this thesis (and everything I have achieved in life). They have loved me so dearly and unconditionally, and yet, allowed me to move half-way across the globe so that I can pursue this academic endeavour. Last but in no ways the least, I can not skip acknowledging the contributions of my sister Smriti (the closest person I have to a second-mother) who has helped me in coping with anxiety and has made herself available to me whenever I have sought her help.

The research done in this thesis was supported by the Indo-US Virtual Networked Joint Center No. IUSSTF/JC-017/2017 and by NSF Award CCF-1955703.

1

Introduction

Decision making under uncertainty is pervasive in today's world and manifests itself in many practical instances such as the decision to rent or buy; constructing the most optimal portfolio in investment; scheduling jobs on a machine(s) etc. A fundamental challenge when it comes to designing any decision making algorithm is the complete lack of information about the future. For instance when deciding whether to rent (or buy) a house, a person may not know how long they will stay at that particular location before they move-out. A longer duration would suggest buying a house while a shorter duration makes renting a prudent choice. Traditionally, theoretical analysis of algorithms under uncertainty have always assumed a pessimistic outlook of the future, i.e the future must hold whatever is worst for our decision-making. While this assumption leads to very robust theoretical guarantees for the worst-case; very little is left to say about the performance of these algorithms in typical real-world instances since the worst-case scenarios might be highly improbable. This problem becomes specifically significant when one considers the plethora of data that is readily available in this age. Typical problem instances arise from a distribution (instead of arising adverserially) and one can expect reasonable forecasts about the future. That being said, we can not completely rely on the forecasts solely since they are not omniscient

(and in certain rare cases, can be completely wrong). In contrast to traditional algorithm design, machine learning (ML) takes a more optimistic approach of trying to predict the future by fitting an appropriate model to past data. A popular line of recent research has been to incorporate ML predictions in the design of online algorithms to improve their performance while preserving (at least partially) the inherent worst-case guarantees. One of the most popular measures of an online algorithm's performance is given by the competitive ratio defined as follows:

Definition 1. *For an online minimization problem P, the competitive ratio of an algorithm $\mathscr{A}$ is said to be α if for all problem instances I, the cost of the algorithm on the instance denoted* $\text{COST}_{\mathscr{A}}(I)$ *satisfies:*

$$\text{COST}_{\mathscr{A}}(I) \leqslant \alpha \cdot \text{COST}_{OPT}(I) + \beta$$

where $\text{COST}_{OPT}(I)$ *denotes the cost of an optimal offline algorithm that knows the entire future in advance and β is a fixed constant.*

The competitive ratio captures the "penalty" an online algorithm has to pay for not knowing the future, since knowing the future would mean performing optimally and having a CR of 1. Our goal is to design learning-augmented algorithms whose competitive ratio gracefully interpolates between optimal offline algorithms if the predictions are accurate – a property called *consistency* – and traditional online algorithms that do not use any predictions – a property called *robustness*.

The above-mentioned approach entails many important questions:

1. What to predict? A problem can have multiple parameters that are dependent on the future, we must investigate what parameters are significant for our decision making.

2. How these predictions should be generated? Typically, any prediction paradigm aims to minimize a certain notion of loss that is related to the mismatch between the

predicted and the actual values. We will see that incorporating the optimization goals of the decision maker in the loss function can lead to provably better performance.

3. Once the predictions are generated, how do we design the decision maker that takes into account these predictions? We want the performance that has a graceful degradation with the accuracy of the predictions.

1.1 Preliminaries

In this section we setup the notations and necessary framework that will be used throughout the thesis.

An online version of a problem is such that the entire information about the problem is not revealed at the beginning but arrives in a piece-by-piece manner. At every instance a piece of information is revealed, an algorithm has to make a decision that can't be changed in the future. In contrast an offline problem is one where the entire input is known to the algorithm at the outset. Examples of online problems include Ski Rental, Paging, Linear Search, Secretary Problem. In addition, many popular NP-hard problems such as Travelling Salesperson or Set Cover have online versions where importance parameters of the problem are revealed in an online fashion.

1.1.1 Rent-or-Buy Problem

In the rent-or-buy problem a player has two options: to buy a particular resource by paying a one-time cost, say $\$B$, or to rent it by paying a smaller recurring cost, say at a rate of \$1 per day. The player does not know the length of usage in advance, and only learns when she is finished using the resource.

Learning to Rent

We denote the feature vector by $x \in \mathfrak{X} \subseteq \mathbb{R}^d$ and the duration length by y. We assume that (x, y) is drawn from an unknown joint distribution $\mathbb{D}$. Given a feature vector x, the goal

of the algorithm is to produce a threshold $\theta(x)$ such that the player rents till time $\theta(x)$ and buys if she is still in need of the resource. The goal is to output a function $\theta(\cdot)$ that minimizes the expected competitive ratio across the distribution. If the distribution $\mathbb{D}$ is known, one can simply compute $\theta^*(\cdot)$ such that $\theta^*(x)$ minimizes the competitive ratio for the conditional distribution $y|x$. However, the distribution $\mathbb{D}$ is not known to the algorithm, and has to be "learned" from training data. The "learning-to-rent" algorithm observes n training samples $(x_i, y_i) \sim \mathbb{D}$, and based on them, generates a function $\theta(\cdot)$ that maps feature vectors x to the wait time.

1.1.2 Online Search Framework

In the ONLINESEARCH framework, there is as an input sequence $\Sigma = \sigma_1, \sigma_2, \ldots$ available offline, and the actual online input is a prefix of this sequence $\Sigma_T = \sigma_1, \sigma_2, \ldots, \sigma_T$, where the length of the prefix T is revealed online. Namely, in each online step $t > 0$, there are two possibilities: either the sequence *ends*, i.e., $T = t$, or the sequence *continues*, i.e., $T > t$. The algorithm must maintain, at all times t, a solution that is feasible for the current sequence, i.e., for the prefix $\Sigma_t = \sigma_1, \ldots, \sigma_t$. The goal is to obtain a solution that is of minimum cost among all the feasible solutions for the actual input sequence Σ_T. Many classical problems fit into this framework (see section 3.2)

Learning to Search

An instance (x, z) of the LEARNTOSEARCH problem is given by a feature $x \in \mathfrak{X}$, and the (unknown) cost of the optimal offline solution $z \in [1, M]$. The two quantities x and z are assumed to be drawn from a joint distribution on $\mathfrak{X} \times [1, M]$. A prediction strategy works with a hypothesis class $\mathscr{F}$ that is a subset of functions $\mathfrak{X} \mapsto [1, M]$ and tries to obtain the best function $f \in \mathscr{F}$ that predicts the target variable z accurately. For notational convenience, we set our target $y = \ln z$, i.e., we try to predict the log-cost of the optimal

solution. [1] Furthermore, let $\mathbb{D}$ denote the input distribution on $\mathfrak{X} \times \mathscr{Y}$, where $\mathscr{Y} = [0, H]$ and $H = \ln M$; i.e., we assume that $(x, y) \sim \mathbb{D}$.

We define a LEARNTOSEARCH algorithm as a strategy that receives a set of m samples $S \sim \mathbb{D}^m$ for *training*, and later, when given the feature set x of a *test* instance $(x, y) \sim \mathbb{D}$ (where y is not revealed to the algorithm), it defines an online algorithm for the ONLINE-SEARCH problem where the optimal cost is e^y.

1.1.3 Online Covering Framework

Online covering is a powerful framework for capturing a broad range of problems in combinatorial optimization. In each online step, a new linear constraint $\mathbf{a} \cdot \mathbf{x} \geqslant b$ is presented to the algorithm, where $\mathbf{x}$ is the vector of variables, $\mathbf{a}$ is a vector of non-negative coefficients, and b is a scalar. The algorithm needs to satisfy the new constraint, and is only allowed to increase the values of the variables to do so. The goal is to minimize an objective function $\mathbf{c} \cdot \mathbf{x}$, where $\mathbf{c}$ is the vector of costs that is known offline.

Online Covering with Multiple Predictions

Our goal will be to obtain a generic algorithm for OCP with multiple suggestions. When the j-th constraint is presented online, the algorithm also receives k suggestions of how the constraint can be satisfied. We denote the s-th suggestion for the j-th constraint by variables $x_i(j, s)$; they satisfy $\sum_{i=1}^{n} a_{ij} x_i(j, s) \geqslant 1$, i.e., all suggestions are feasible.

To formally define the *best* suggestion, we say that a solution $\{x_i : i \in [n]\}$ is *supported* by the suggestions $\{x_i(j) : i \in [n], j \in [m]\}$ if $x_i \geqslant x_i(j)$ for all $j \in [m]$. Using this definition, we consider below two natural notions of the best suggestion that we respectively call the *experts setting* and the *multiple predictions setting*.

[1] Note that predicting the log-cost of $\text{OPT}(T)$ is equivalent to predicting the input length T; when multiple input lengths might have the same optimal cost, we can just pick the longest one.

The Experts Setting. In the experts setting, there are k experts, and the s-th suggestion for each constraint comes from the same fixed expert $s \in [k]$ (say some fixed ML algorithm or a human expert). The online algorithm is required to be competitive with the *best* among these k experts[2]. To formalize this, we define the benchmark:

$$\text{STATIC} = \min_{s\in[k]} \sum_{i=1}^{n} c_i \cdot \max_{j\in[m]} x_i(j,s).$$

Note that $\{\max_{j\in[m]} x_i(j,s) : i \in [n]\}$ is the *minimal* solution that is supported by the suggestions of expert s; hence, we define the cost of the solution proposed by expert s to be the cost of this solution.

The Multiple Predictions Setting. In the multiple predictions setting, we view the set of k suggestions in each step as a bag of k predictions (without indexing them specifically to individual predictors or experts) and the goal is to obtain a solution that can be benchmarked against the best of these suggestions in each step. Formally, our benchmark is the minimum-cost solution that is supported by at least one suggestion in each online step:

$$\text{DYNAMIC} = \min_{\hat{\mathbf{x}}\in\hat{X}} \sum_{i=1}^{n} c_i \cdot \hat{x}_i, \text{ where}$$

$$\hat{X} = \{\hat{\mathbf{x}} : \forall i \in [n], \forall j \in [m], \exists s \in [k], \hat{x}_i \geqslant x_i(j,s)\}.$$

1.2 Summary of Results

1.2.1 Rent-or-buy

Our goal is to design a learning-to-rent algorithm with an expected competitive ratio of $(1+\varepsilon)$, and analyze the dependence of the number of samples n on the value of ε.

[2] This is similar to the experts model in online learning, hence the name.

– By making mild Lipchitz assumptions on the joint distribution (x, y), we design a $(1+\varepsilon)$ accurate algorithm. However this has a sample complexity of $O(\frac{1}{\varepsilon^{O(d)}})$ where d is dimension of the sample space. We show that we can not avoid an exponential dependence on d without making further assumptions.

– To reduce the sample complexity, we make an assumption that the binary variable $y \geqslant B$ is PAC-learnable by a hypothesis class of VC Dimension d, and give a $1+\varepsilon$ competitive algorithm using just $O(\frac{d}{\varepsilon^2})$ samples. Furthermore, we exploit the specific structure of the problem to improve the sample complexity of the training set to remove the dependence on d entirely (although at a slightly worse dependence on ε).

– We also consider a noisy model where the labels in the training set are noisy, i.e : labels for a certain fraction of the input distribution are flipped adverserially. We give a noise tolerant algorithm for the learning-to-rent problem with a competitive ratio of $1+3\sqrt{p}$, where p is the mis-classification error of a noise tolerant binary classifier. We complement this bound by showing that for a noise level of η, the best competitive ratio achievable is $1+\frac{\sqrt{\eta}}{2}$.

– For robustness, we show that none of our algorithms has competitive ratios any worse than $1+\frac{1}{\varepsilon}$ in the adversarial setting.

1.2.2 Online Search

– For the LEARNTOSEARCH problem, we show a sample complexity bound of $O\left(\frac{H \cdot d}{\varepsilon}\right)$ for obtaining a competitive ratio of $1+\varepsilon$, where H and d respectively represent the log-range of the optimal cost and a measure of the expressiveness of the function class $\mathscr{F}$ called its pseudo-dimension.

– We also extend this result to the so-called agnostic setting, where the function class $\mathscr{F}$ is no longer guaranteed to contain an exact function f that maps x to z, rather the

competitive ratio is now in terms of the *best* function in this class that approximates f. We also prove nearly matching lower bounds for our sample complexity bounds in the two models.

– Our framework can also be extended to the setting where the offline optimal solution is hard to compute, but there exists an algorithm with competitive ratio c given the *cost* of optimal solution. In that case our algorithms gives a competitive ratio $c(1+\varepsilon)$, which can still be better than the competitive ratio without predictions.

Furthermore, we give three examples of classic problems – ski rental with multiple options, online scheduling, and online bin packing – to illustrate the applicability of the ONLINESEARCH framework.

1.2.3 Online Covering

– We design an OCP algorithm whose cost is $O(\log k)$ competitive with respect to the best suggested solution. We give a lower bound of $\Omega(\log k)$ by only using binary (0/1) coefficients and unit cost for each variable, which implies that the lower bound holds even for the special case of the unweighted set cover problem.

– Using standard techniques, we observe that the OCP algorithm can be *robustified*, i.e., along with being $O(\log k)$-competitive against the best suggested solution, the algorithm can be made $O(\alpha)$-competitive against the optimal solution where α is the competitive ratio of the best online algorithm (without predictions).

– Next, we generalize the online covering framework by introducing *box*-type constraints. We demonstrate that our techniques and results from online covering extend to this more general setting.

We use these formulations for solving online set cover, online caching and online facility location problems – in the multiple predictions setting.

1.3 Thesis Roadmap

In chapters 2 and 3 we examine how we can leverage the problem structure in generating suitable predictions for online algorithms. This is done using a classification based approach for Rent-or-Buy in chapter 2 and a regression based approach for ONLINESEARCH in chapter 3. In Chapter 4 we examine the scenario where we have multiple predicted solutions, and we try to be competitive against the *best* solution. Chapter 5 contains the concluding remarks and discussions on future research.

2

Learning-Augmented Algorithms for Rent-or-Buy

2.1 Introduction

The *rent-or-buy* (a.k.a. *ski rental*) problem is considered a classical problem of optimization under uncertainty. In this problem, an algorithm is faced with one of two choices: a small recurring (rental) cost, or a large (buying) cost that has to be paid once but no cost thereafter. This choice routinely arises in our daily lives such as in the decision to rent or buy a house, corporate decisions to rent or buy data centers, expensive equipment, and so on. Naturally, the optimal choice depends on the duration of use, a longer duration justifying the decision to buy instead of renting. But, this is where the uncertainty lies: the length of use is often not known in advance. We formally define this problem next.

2.1.1 Problem Definition

In the ski rental problem, a skier has two options: to buy skis at a one time cost of (say) $\$B$ or to rent them at a cost of \$1 per day. The skier does not know the length of the ski season in advance, and only learns it once the season ends. Note that if the length of the season were known, then the optimal policy is to buy at the beginning of the season if it lasts longer than B days, and rent every day if it is shorter. But, in the absence of this information, an

algorithm has to decide the duration of renting skis before buying them.

2.1.2 Background

The ski rental problem is perhaps the most fundamental, and structurally simplest, of all problems in online algorithms, and has been widely studied in many contexts (see, e.g., Karlin et al. (1994, 2003); Lotker et al. (2008); Khanafer et al. (2013); Kodialam (2014)), including that of online algorithms with ML predictions Purohit et al. (2018); Gollapudi and Panigrahi (2019a). It is well-known that the best competitive ratio achievable by a deterministic algorithm for this problem is 2 (e.g., Karlin et al. (1988)), and that by a randomized algorithm is $\frac{e}{e-1}$ (e.g., Karlin et al. (1994)). The ski-rental problem Karlin et al. (1994); Lotker et al. (2008); Khanafer et al. (2013); Kodialam (2014), and variants such as TCP acknowledgment Karlin et al. (2003), the parking permit problem Meyerson (2005a), snoopy caching Karlin et al. (1988), etc. model the fundamental difficulty in decision making under uncertainty in many situations.

2.1.3 Our Contributions

We consider a "learning" version of Ski-Rental problem where the length of the ski-season is not adversarial but instead arises from an unknown distribution. We show tight sample complexity bounds to obtain a solution that is within a $1+\varepsilon$ factor of the best possible solution for that distribution. Next, we consider a PAC-learning framework where we use a black-box classifier that predicts whether the number of ski-days is greater than the threshold for buying (or not) and give a $1+2\cdot\sqrt{\varepsilon}$ competitive algorithm if the mis-classification error is ε. We further demonstrate that these results are noise-tolerant. Finally, we leverage the specific structure of the rent-or-buy problem to open the black-box learner and design a special margin-based classifier. By using this new prediction pipeline, we are able to further improve our sample complexity bounds.

2.2 Learning Framework

For notational convenience, we consider a continuous version of the ski rental problem, where the buying cost is \$1, and the length of the ski season is denoted by y. (The assumption on the buying cost is w.l.o.g. by appropriate scaling.) Therefore, the optimal offline solution is to buy at the outset when $y \geqslant 1$ and rent throughout when $y < 1$. We also denote the feature vector by $x \in \mathbb{R}^d$ (e.g., weather predictions, skier behavior, etc.) and assume that (x, y) is drawn from an unknown joint distribution $\mathbb{D}$. Given a feature vector x, the goal of the algorithm is to produce a threshold $\theta(x)$ such that the skier rents till time $\theta(x)$ and buys at that point if the ski season is longer. We call $\theta(x)$ the *wait time* of the algorithm.

If the distribution $\mathbb{D}$ were known to the algorithm, then for each input x, it can compute the conditional distribution $y|x$ and solve the resulting *stochastic* ski rental problem, i.e., where the input is drawn from a given distribution. It is well known that the optimal strategy in this case can be described by a fixed wait time that we denote as $\theta^*(x)$.

Of course, in general, the distribution $\mathbb{D}$ is not known to the algorithm, and has to be "learned" from training data. The "learning-to-rent" algorithm observes n training samples $(x_i, y_i) \sim \mathbb{D}$, and based on them, generates a function $\theta(x)$ that maps feature vectors x to the wait time. The (expected) competitive ratio of the algorithm is given by:

$$\text{CR}(\theta, \mathbb{D}) = \mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta(x), y)] \tag{2.1}$$

$$\text{where } g(\theta(x), y) = \begin{cases} \frac{y}{\min\{y,1\}} & \text{when } y < \theta(x) \\ \frac{1+\theta(x)}{\min\{y,1\}} & \text{when } y \geqslant \theta(x). \end{cases} \tag{2.2}$$

The goal of the learning-to-rent algorithm is to output a function $\theta(\cdot)$ that minimizes CR in Eq. (2.1). Since the ideal strategy is to output the function $\theta^*(\cdot)$, we measure the performance of the algorithm as the ratio between $\text{CR}(\theta, \mathbb{D})$ and $\text{CR}(\theta^*, \mathbb{D})$.

Definition 2. *A learning-to-rent algorithm A with threshold function $\theta(\cdot)$ is said be (ε, δ)-accurate with n samples, if for any distribution $\mathbb{D}$, after observing n samples, we have the*

following guarantee with probability at least $1-\delta$*:*

$$\mathrm{CR}(\theta, \mathbb{D}) \leqslant (1+\varepsilon) \cdot \mathrm{CR}(\theta^*, \mathbb{D}). \tag{2.3}$$

If we say that an algorithm is $(1+\varepsilon)$*-accurate, we mean Eq.* (2.3) *holds for some fixed constant* δ*.*

The additional parameter $\mathbb{D}$ can be dropped when the distribution is clear from the context.

2.3 Our Approach

In this section we give the main ideas about our algorithms. For arbitrary lipchitz distributions, discretize the sample space using an ε-net. Then, for each cell in the ε-net, we have one of two cases. Either, there are sufficiently many samples to estimate the conditional distribution $y \mid x$. Or, a baseline online algorithm can be used for the cell if it has very few samples.

Our next goal is to improve the dependence on d since the number of features in a typical setting can be rather large, which would make the previous algorithm prohibitively expensive. To this end, we use a PAC learning approach to address the problem. Since the optimal ski rental policy exhibits threshold behavior (rent throughout if $y < B$ and buy at the outset if $y \geqslant B$), we treat the underlying learning problem as a classification task. In particular, we introduce an auxiliary binary variable z that captures the two regimes for the optimal ski rental policy:

$$z = \begin{cases} 1 & \text{if } y \geqslant B \\ 0 & \text{if } y < B \end{cases}$$

We then show that if z belongs to a concept class that is (ε, δ) PAC-learnable from x, then we can obtain a learning-to-rent algorithm that achieves a competitive ratio of $(1+2\sqrt{\varepsilon})$ with probability $1-\delta$. This implies, for instance, that if there were a linear classifier for z, then the number of required training samples n to obtain a $(1+\varepsilon)$ competitive algorithm

can be decreased from exponential to linear in d, specifically $O(d/\varepsilon^2)$. Finally, our most significant idea is to modify the loss function in the learning algorithm to incorporate the optimization objective. In particular, we observe that the classification error is almost entirely due to samples close to the threshold, but for values of y close to B, mis-classifying z does not cost us significantly in the ski rental objective. This allows us to create an artificial margin around the classification boundary and discard all samples that appear in this margin. Using this improvement, we can improve the sample complexity of the training set to remove the dependence on d entirely (although at a slightly worse dependence on ε)

2.4 A General Learning-to-Rent Algorithm

As described in the introduction, it is natural (and required) to assume that the joint distribution $\mathbb{D}$ on (x,y) is **Lipschitz** in the sense that similar feature vectors x imply similar conditional distributions $y|x$. In this section, our main contribution is to design a learning-to-rent algorithm under this minimal assumption.

First, we give the precise definition of the Lipschitz property we require. In particular, we measure distances between distributions using the *earth mover distance* (EMD) metric.

Definition 3. *For probability distributions $\mathbb{X},\mathbb{Y}$ over $\mathbb{R}^d$,*

$$\text{EMD}(\mathbb{X},\mathbb{Y}) = \min_{\mathbb{D}:\mathbb{D}|x=\mathbb{X},\mathbb{D}|y=\mathbb{Y}} \left(\mathbb{E}_{(x,y)\sim\mathbb{D}}[\|x-y\|]\right).$$

The joint distribution $\mathbb{D}$ above is such that its marginals with respect to y and x are equal to $\mathbb{X}$ and $\mathbb{Y}$ respectively. We now define the Lipschitz property using EMD as the distance measure between distributions.

Definition 4. *A joint distribution on $(x,y) \in \mathbb{R}^d \times \mathbb{R}^+$ is said to be L-Lipschitz iff for all $x_1,x_2 \in \mathbb{R}^d$, the marginal distributions $\mathbb{Y}_1 = y|x_1$, $\mathbb{Y}_2 = y|x_2$ satisfy $\text{EMD}(\mathbb{Y}_1,\mathbb{Y}_2) \leqslant L\cdot\|x_1-x_2\|_2$.*

Now we are ready to state our main result in this section:

Theorem 1. *For the learning-to-rent problem, if $x \in [0,1]^d$, and the joint distribution (x,y) is L-Lipschitz, then there exists an algorithm that uses $n = \left(\frac{L\sqrt{d}}{\varepsilon}\right)^{O(d)}$ samples and is $(1+\varepsilon)$-accurate with high probability.*[1] [2]

Algorithm 1 Outputs θ_A for a given distribution on y

Query $\left(\frac{\delta}{\varepsilon^6}\right)$ samples for some constant $\delta > 0$.
Initialize array l of length $\frac{1}{\varepsilon^2}$
Let $\ell[\theta] \leftarrow$ average of $g(\theta, y)$ over all samples y.
return $\theta_A \leftarrow \arg\min_{\theta \in [\varepsilon, 1/\varepsilon], \theta/\varepsilon \in \mathbb{N}} \ell[\theta]$.

2.4.1 Algorithm for a fixed feature

Let us first consider a warm-up example where we have a fixed x and only consider the conditional distribution $y|x$ (See Algorithm 1). In this case, it is natural to optimize θ over the empirical samples of y. However, if we don't put any constraint on θ, the competitive ratio for a sample y can be unbounded (this can happen when θ is close to 0 or very large), which might hurt generalization. We solve this problem by proving that it suffices to consider θ in the range $[\varepsilon, 1/\varepsilon]$ in order to get an $(1+\varepsilon)$-accurate solution. (See Lemma 2).

For this special case, we have the following result:

Theorem 2. *If a learning-to-rent problem has only one possible input x, then there exists an algorithm requiring $O\left(\frac{\delta}{\varepsilon^6}\right)$ samples that achieves $(1+\varepsilon)$ accuracy with probability $\geqslant 1 - O\left(\frac{e^{-\Omega\left(\frac{\delta}{\varepsilon}\right)}}{\varepsilon^2}\right)$.*

Let $\mathbb{D}$ be the distribution of y, and θ^* be the optimal threshold for this distribution and $f^*_{\mathbb{D}}$ is the optimal expected competitive ratio. We first show that it suffices to get a threshold that is not much larger than θ^*:

[1] with probability exceeding $1 - \varepsilon^{\Omega(d)}$

[2] The quantity ε is considered to be small ($\leqslant 0.01$) throughout the analysis

Algorithm 2 Outputs $\theta_A(x)$ for multi-dimensional x

Divide the hyper-cube $[0,1]^d$ into sub-cubes of side length $\frac{\varepsilon^3}{64L\sqrt{d}}$ each. The number of such cubes is $N = \left(\frac{64L\sqrt{d}}{\varepsilon^3}\right)^d$. Index the cubes by i, where $1 \leqslant i \leqslant N$.

Query $\Pi = \left(\frac{1024L\sqrt{d}}{\varepsilon^6}\right)^{2d}$ samples, and let $I_\varepsilon = [\varepsilon, 1/\varepsilon]$.

Set threshold $\tau = \left(\frac{64L\sqrt{d}}{\varepsilon^8}\right)^d$.

for each sub-cube C_i:

  if the number of samples from the sub-cube exceeds τ

  then

    Compute $\theta_i \leftarrow \arg\min_{\theta \in I_\varepsilon, \theta/\varepsilon \in \mathbb{N}} \mathbb{E}_{(x,y):x\in C_i}[g(\theta,y)]$.

    For all $x \in C_i$: **return** $\theta_A(x) \leftarrow \theta_i$.

  else

    For all $x \in C_i$: **return** $\theta_A(x) \leftarrow 1$.

Lemma 1. *Let the length of the ski-renting season $y \sim \mathbb{D}$, then:*

$$\mathrm{CR}(\theta^* + \varepsilon, \mathbb{D}) \leqslant (1+\varepsilon) f^*_{\mathbb{D}}$$

where $f^*_{\mathbb{D}} = \mathrm{CR}(\theta^*, \mathbb{D})$ is the optimal value and the optimal threshold $\theta^* = \arg\min_{\theta \in \mathbb{R}^+} \mathrm{CR}(\theta, \mathbb{D})$.

Proof. We compare the competitive ratio at different values of y. Recall that :

$$g(\theta, y) = \begin{cases} \frac{(1+\theta)}{\min\{1,y\}} & \text{if } y \geqslant \theta \\ \frac{y}{\min\{1,y\}} & \text{otherwise} \end{cases}$$

When $y \leqslant \theta^*$ then both thresholds will lead to the same cost and $g(.,y)$ remains unchanged.

For $\theta^* + \varepsilon > y > \theta^*$ we have

$$\frac{g(\theta^* + \varepsilon, y)}{g(\theta^*, y)} = \frac{y}{1+\theta^*} \leqslant 1.$$

Finally, for $y > \theta^* + \varepsilon$, we have

$$\frac{g(\theta^* + \varepsilon, y)}{g(\theta^*, y)} = \frac{1+\theta^*+\varepsilon}{1+\theta^*} \leqslant (1+\varepsilon).$$

Since the ratio is bounded above by $1+\varepsilon$ for all y, after taking the expectation we have

$$\mathbb{E}_{y\sim\mathbb{D}}[g(\theta^*+\varepsilon,y)] \leqslant (1+\varepsilon)\cdot\mathbb{E}_{y\sim\mathbb{D}}[g(\theta^*,y)].$$

□

The next lemma shows that without loss of generality we only need to consider thresholds in the range $[\varepsilon, 1/\varepsilon]$:

Lemma 2. *Let* $f_{\mathbb{D}}^{\varepsilon} = \min_{\theta\in\left[\varepsilon,\frac{1}{\varepsilon}\right]} \mathrm{CR}(\theta,\mathbb{D})$ *then:*

$$f_{\mathbb{D}}^{\varepsilon} \leqslant f_{\mathbb{D}}^{*}(1+\varepsilon),$$

where $f_{\mathbb{D}}^{*} = \mathrm{CR}(\theta^*,\mathbb{D})$ *is the optimal value, the optimal threshold being* $\theta^* = \arg\min_{\theta\in\mathbb{R}^+} \mathrm{CR}(\theta,\mathbb{D})$.

Proof. Let $\theta^{\varepsilon} = \arg\min_{\theta\in\left[\varepsilon,\frac{1}{\varepsilon}\right]}$ be the optimal threshold within the range $[\varepsilon,1/\varepsilon]$. We consider different cases for the optimal threshold (without constraints) θ^*.

Case I: When $\theta^* \in \left[\varepsilon,\frac{1}{\varepsilon}\right]$ then clearly we have $\theta^* = \theta^{\varepsilon}$.

Case II : $\theta^* < \varepsilon$, in this case we show that choosing $\theta = \theta^* + \varepsilon$ is good enough: by Lemma 1, we have that $\theta^*+\varepsilon \in \left[\varepsilon,\frac{1}{\varepsilon}\right]$ and, $f_{\mathbb{D}}^{\varepsilon} \leqslant \mathrm{CR}(\theta^*+\varepsilon,\mathbb{D}) \leqslant (1+\varepsilon)f_{\mathbb{D}}^{*}$.

Case III : $\theta^* > \frac{1}{\varepsilon}$, in this case we show that choosing $\theta = 1/\varepsilon$ is good enough. When $y \leqslant 1/\varepsilon$, then $g(1/\varepsilon,y) \leqslant g(\theta^*,y)$. When $y > 1/\varepsilon$, then $\frac{g(1/\varepsilon,y)}{g(\theta^*,y)} \leqslant \frac{1/\varepsilon+1}{y} \leqslant \frac{1/\varepsilon+1}{1/\varepsilon} = 1+\varepsilon$.

Hence, $f_{\mathbb{D}}^{\varepsilon} \leqslant \mathbb{E}_{y\sim\mathbb{D}}[g(1/\varepsilon,y)] \leqslant (1+\varepsilon)f_{\mathbb{D}}^{*}$. □

Next we show how to estimate the expected competitive ratio using samples from the distribution.

Lemma 3. *Given a fixed* $\theta \in \left[\varepsilon,\frac{1}{\varepsilon}\right]$*, by taking* $\frac{\delta}{\varepsilon^4}$ *samples of* $y\sim\mathbb{D}$*, the quantity* $\mathbb{E}_{y\sim\mathbb{D}}[g(\theta,y)]$ *can be estimated to a multiplicative accuracy of* ε *with probability* $1-e^{-\frac{2\delta}{\varepsilon}}$.

Proof. Note that when $\theta \in \left[\varepsilon,\frac{1}{\varepsilon}\right]$ then $g(\theta,y)$ is bounded above by $\frac{1}{\varepsilon}+1$, therefore the random variable $g(\theta,y)$ has a variance σ^2 bounded above by $\frac{1}{\varepsilon^2}$.

Let $\text{CR}(\theta,\mathbb{D}) = \mathbb{E}_{y\sim\mathbb{D}}[g(\theta,y)]$ be the true mean of the distribution and $\widehat{\text{CR}}(\theta,\mathbb{D})$ denotes the estimate that we have obtained by taking $\frac{\delta}{\varepsilon^4}$ samples. Also, any estimate of $g(\theta,y)$ is from a distribution whose mean is $\text{CR}(\theta,\mathbb{D})$ and is bounded inside the range $[1,1+\frac{1}{\varepsilon}]$. Therefore, taking $\frac{\delta}{\varepsilon^4}$ samples and by Hoeffding's Inequality Hoeffding (1963), we claim that :

$$\mathbb{P}\left[\widehat{\text{CR}}(\theta,\mathbb{D}) - \text{CR}(\theta,\mathbb{D}) > t\right] \leqslant exp\left(-\frac{2\delta t}{\varepsilon^2}\right).$$

Setting $t = \varepsilon$ and using the fact that $\text{CR}(\theta,\mathbb{D}) \geqslant 1$, we get that with probability: $1 - e^{-\frac{2\delta}{\varepsilon}}$,

$$\widehat{\text{CR}}(\theta,\mathbb{D}) \leqslant (1+\varepsilon)\text{CR}(\theta,\mathbb{D}).$$

□

Finally, we are ready to prove Theorem 2:

Proof of Theorem 2. The algorithm simply involves dividing the segment $\left[\varepsilon,\frac{1}{\varepsilon}\right]$ into small intervals of ε width. This would give us at most $1/\varepsilon^2$ intervals.(refer to Algorithm 1) For each interval $[\theta_0-\varepsilon,\theta_0]$ we use the $\frac{\delta}{\varepsilon^6}$ samples at $\theta=\theta_0$ to calculate $\widehat{\text{CR}}(\theta_0,\mathbb{D})$. We output the θ_0 that has the minimum $\widehat{\text{CR}}(\theta_0,\mathbb{D})$ over all such intervals.

By Lemma 3 we know that our estimate is within a $(1+\varepsilon)$ multiplicative factor of the true $\text{CR}(\theta_0,\mathbb{D})$ with probability: $1-e^{-\frac{2\delta}{\varepsilon}}$. Since there are at most $\frac{1}{\varepsilon^2}$ such θ_0: by a simple union bound, we claim that all our estimates on the competitive ratio are $(1+\varepsilon)$ multiplicative factor of the true $\text{CR}(\theta_0,\mathbb{D})$ with probability : $1-\left(\frac{e^{-\frac{2\delta}{\varepsilon}}}{\varepsilon^2}\right)$. Also lemma 1 tells us that $\text{CR}(\theta_0,\mathbb{D})$ is within a $(1+\varepsilon)$ factor of $\text{CR}(\theta,\mathbb{D})$ for all $\theta\in[\theta_0-\varepsilon,\theta_0]$. Therefore, by taking the minimum over all θ_0 : we are within a $(1+2\varepsilon+\varepsilon^2)$ factor of $\min_{\theta\in\left[\varepsilon,\frac{1}{\varepsilon}\right]}\text{CR}(\theta,\mathbb{D})$. Finally, we invoke lemma 2 to claim that our value is within a $(1+4\varepsilon)$ (for $\varepsilon<0.4$) multiplicative factor of f^*. Repeating the above analysis with $\varepsilon'=\frac{\varepsilon}{4}$, we achieve $(1+\varepsilon')$ accuracy using $\frac{256\delta}{\varepsilon'^4}$ samples with probability: $1-16\left(\frac{e^{-8\delta/\varepsilon'}}{\varepsilon'^2}\right)$. □

2.4.2 Algorithm for Multi-dimensional Distribution

To go from a single x to the whole distribution, the main idea is to discretize the domain of x using an ε-net for small enough ε.[3] For each cell in the ε-net, we show that if there are enough samples in the training set from that cell, then we can estimate the conditional probability $y|x$ to a sufficient degree of accuracy for the optimization loss to be bounded by $1+\varepsilon$. On the other hand, if there are too few samples, then the probability density in the cell is small enough that it suffices to use a worst case online algorithm for all test data in the cell. (The formal algorithm is given in Algorithm 2.)

Lemma 4. *Given two distributions $\mathbb{D}_1, \mathbb{D}_2$ such that EMD$(\mathbb{D}_1, \mathbb{D}_2) \leqslant \Delta$, then:*

$$\mathbb{E}_{y\sim\mathbb{D}_1}[g(\theta+\varepsilon, y)] \leqslant (1+\varepsilon)\left(1+\frac{\Delta}{\varepsilon^2}\right)\mathbb{E}_{y\sim\mathbb{D}_2}[g(\theta, y)], \text{ for any } \theta \in \mathbb{R}^+.$$

Proof. Let $p_i(y_0)$ be the probability that $y = y_0$ for distribution $\mathbb{D}_i$. For $y \leqslant \theta+\varepsilon$: $\frac{g(\theta+\varepsilon,y)}{g(\theta,y)} \leqslant 1$. Also, when $y > \theta+\varepsilon$ then, $\frac{g(\theta+\varepsilon,y)}{g(\theta,y)} = \frac{1+\theta+\varepsilon}{1+\theta} \leqslant (1+\varepsilon)$.

Let us begin at distribution $\mathbb{D}_2$, and there be an adversary who wants to increase the expectation $\mathbb{E}_y[g(\theta+\varepsilon, y)]$ by shifting some probability mass and thereby changing the distribution. However the adversary cannot change the distribution drastically (which is where the EMD comes into play), the total earth mover distance between the new and old distribution can be at most Δ.

The figure 2.1 shows the different values of $g(\theta, y)$ and $g(\theta+\varepsilon, y)$ in the regions where $y \leqslant \theta$, $y \in (\theta, \theta+\varepsilon]$ and $y > \varepsilon+\theta$. Note that the difference $g(\theta+\varepsilon, y) - g(\theta, y)$ is greatest when $y > \varepsilon+\theta$. Shifting any probability mass within the regions $y < \theta$ or $y > \theta+\varepsilon$ does not increase the quantity $\frac{g(\theta+\varepsilon,y)}{g(\theta,y)}$. If we shift some probability mass from $y_1 \in [\theta, \theta+\varepsilon]$ to $y_2 > \theta+\varepsilon$, the increase in $\frac{g(\theta+\varepsilon,y_2)}{g(\theta,y_1)}$ is upper bounded by $1+\varepsilon$. Note that we can shift as

[3] The ε in the ε-net is not the same as the accuracy parameter ε. We are overloading ε in this description because the reader may be familiar with the term ε-net; in the formal algorithm (Algorithm 2), we avoid this overloading.

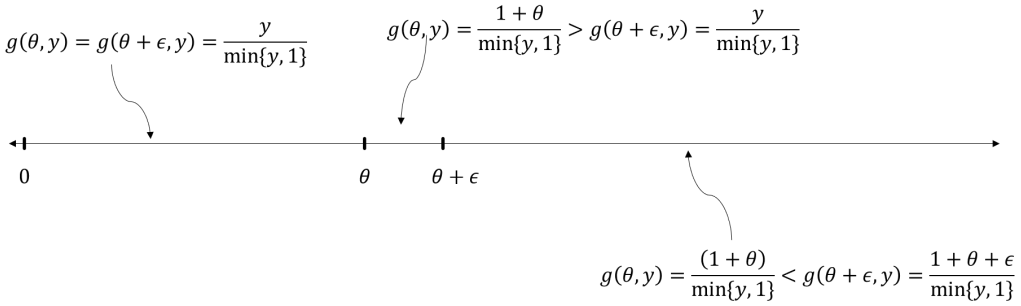

FIGURE 2.1: Value of $g(\theta,y)$ in different regimes of y

much mass as we want from $y_1 = \theta+\varepsilon-\tau$ to $y_2 = \theta+\varepsilon+\tau$ for $\tau \to 0^+$.

The maximum change occurs when we move from $y_1 < \theta$ to $y_2 > \theta+\varepsilon$, then $\frac{g(\theta+\varepsilon,y_2)}{g(\theta,y_1)} = \left(\frac{\min\{1,y_1\}}{y_1} \cdot \frac{(1+\theta+\varepsilon)}{\min\{1,y_2\}}\right) \leqslant \left(\frac{(1+\theta+\varepsilon)}{\min\{1,y_2\}}\right)$. However, the maximum probability mass that can be moved is upper bounded by $\frac{\Delta}{y_2-y_1}$ (Since we know that $\mathbb{D}_1$ and $\mathbb{D}_2$ differ by Δ).

Thus the upper bound we obtain is,

$$\begin{aligned}
\frac{\mathbb{E}_{y\sim\mathbb{D}_1}[g(\theta+\varepsilon,y)]}{\mathbb{E}_{y\sim\mathbb{D}_2}[g(\theta,y)]} &\leqslant (1+\varepsilon) + \max_{y_1\in[0,\theta),y_2>\theta+\varepsilon}\left(\frac{(1+\theta+\varepsilon)}{\min\{1,y_2\}} \times \frac{\Delta}{y_2-y_1}\right) \\
&= (1+\varepsilon) + \left(\frac{1+\theta+\varepsilon}{\theta+\varepsilon} \times \frac{\Delta}{\varepsilon}\right) \\
&\leqslant (1+\varepsilon) + (1+\varepsilon)\frac{\Delta}{\varepsilon^2} \\
&= (1+\varepsilon)\left(1+\frac{\Delta}{\varepsilon^2}\right)
\end{aligned}$$

□

As a corollary, by linearity of expectation we know if many distributions are close, then their optimal solutions are also close:

Corollary 1. *Let $\mathbb{D}_1,\mathbb{D}_2$ be two joint distributions on (X,Y) such that they have the same support S on X. If $\forall x_i,x_j \in S$, $\mathbb{D}_i = Y \mid (X=x_i), \mathbb{D}_j = Y \mid (X=x_j)$ satisfies $EMD(\mathbb{D}_i,\mathbb{D}_j) \leqslant \Delta$ then:*

$$\mathbb{E}_{(x,y)\sim\mathbb{D}_1}[g(\theta+\varepsilon,y)] \leqslant (1+\varepsilon)\left(1+\frac{\Delta}{\varepsilon^2}\right)\mathbb{E}_{(x,y)\sim\mathbb{D}_2}[g(\theta,y)]$$

And,

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}_1}[g(\theta,y)] \leqslant (1+\varepsilon)\left(1+\frac{\Delta}{\varepsilon^2}\right)\min_{\theta}\mathbb{E}_{(x,y)\sim\mathbb{D}_2}[g(\theta,y)]$$

We now give the proof of Theorem 1 to show the sample complexity to obtain a $1+\varepsilon$ learning-to-rent algorithm:

Proof. Let us focus on a certain sub-cube C_i, we will break them into two cases: one where the sub-cube gets enough samples and one where the sub-cube does not get enough samples.

CASE I: Let's say that we met the threshold and got over $\frac{1}{\varepsilon^{8d}}$ samples in C_i. Let $\mathbb{D}_i$ be the true conditional distribution of Y when $X = x$ lies in C_i. Clearly, when we are sampling Y where X lies inside C_i our estimate might be a from a different distribution $\hat{\mathbb{D}}_i$.

But both these distributions are from a linear combinations of conditional distributions $Y \mid (X = x)$ over $x \in C_i$. Using algorithm 1 we get a θ_i for C_i and using the result from Theorem 2 (with $\delta = \frac{1}{\varepsilon^{8d-6}}$), and union bound over all the cubes, we can claim that with a very high probability: $\geqslant 1 - O\left(\varepsilon^{\Omega(d)}\right)$, it satisfies:

$$\forall i\, \mathbb{E}_{y\sim\hat{\mathbb{D}}_i}[g(\theta_i,y)] \leqslant \left(1+\frac{\varepsilon}{3}\right)\cdot\min_{\theta}\mathbb{E}_{y\sim\hat{\mathbb{D}}_i}[g(\theta,y)] \tag{2.4}$$

Using Corollary 1 we have the following

$$\min_{\theta}\mathbb{E}_{y\sim\mathbb{D}_i}[g(\theta,y)] \leqslant \left(1+\frac{\varepsilon}{4}\right)\left(1+\frac{16\Delta}{\varepsilon^2}\right)\min_{\theta}\mathbb{E}_{y\sim\hat{\mathbb{D}}_i}[g(\theta,y)] \tag{2.5}$$

Since for any $x, y \in C$, we have $\|x-y\|_2 \leqslant \frac{\varepsilon^3}{64L}$, therefore using the Lipchitz assumption, we have $\Delta \leqslant \frac{\varepsilon^3}{64}$. Hence,

$$\min_{\theta}\mathbb{E}_{y\sim\hat{\mathbb{D}}_i}[g(\theta,y)] \leqslant \left(1+\frac{\varepsilon}{3}\right)\min_{\theta}\mathbb{E}_{y\sim\mathbb{D}_i}[g(\theta,y)]. \tag{2.6}$$

Using Theorem 2, and for $\varepsilon < 0.1$,

$$\mathbb{E}_{y\sim\mathbb{K}_i}[g(\theta_i,y)] \leqslant \left(1+\frac{3\varepsilon}{4}\right)\min_{\theta}\mathbb{E}_{y\sim\mathbb{D}_i}[g(\theta,y)]. \tag{2.7}$$

CASE II: When C_i does not have enough samples to meet the threshold and we set $\theta_A(x) = 1$ for all $x \in C_i$. In this case, we have that $g(\theta_A(x), y) = g(1, y) \leqslant 2$.

We will see now that the second case occurs with a very small probability. Let $P[x \in C_i]$ be denoted as p_i and let $\hat{p}_i$ be our empirical estimation of p_i. By Hoeffding's bound,

$$\mathbb{P}[\|p_i - \hat{p}_i\| \geqslant t] \leqslant 2e^{-2\Pi \cdot t^2}.$$

where $\Pi = \left(\frac{1024L\sqrt{d}}{\varepsilon^6}\right)^{2d}$ is the number of samples we took. If we set $t = \frac{\varepsilon^{4d}}{(1024L\sqrt{d})^d}$, we have: $\|p_i - \hat{p}_i\| < \frac{\varepsilon^{4d}}{(1024L\sqrt{d})^d}$, with probability : $\geqslant 1 - 2exp(-\frac{2}{\varepsilon^{4d}})$. By carrying a simple union bound over all such i, we show that the above relation holds true for all C_i with probability:

$$1 - N \cdot 2e^{-\frac{2}{\varepsilon^{4d}}}.$$

Using simple inequalities like $e^{-x} < \frac{1}{x^2}$ for $x > 0$ we can show that this probability is greater than $\alpha = 1 - O(\varepsilon^{\Omega(d)})$.

Let a cube be termed **good** if it has the threshold satisfied and **bad** otherwise. Also, $C(x)$ denotes the cube which contains x and n_i is the number of samples lying inside cube C_i. Let $\mathbb{V}$ denote the discrete distribution of x over the cubes. The probability $p_i = \mathbb{P}_{x \sim \mathbb{V}}[x \in C_i]$ that an x chosen from $\mathbb{V}$ will lie in C_i is estimated as $\hat{p}_i = \frac{n_i}{\Pi}$ and as shown above, with

probability $\geqslant 1-\alpha$: $\|p_i - \hat{p}_i\| < \frac{\varepsilon^{4d}}{(1024L\sqrt{d})^d}$ We obtain:

$$\begin{aligned}
\mathbb{P}_{x\sim\mathbb{V}}[C(x) \text{ is good}] &= \sum_{C_i \text{ is good}} p_i \\
&\geqslant \sum_{C_i \, is \, good} \frac{n_i}{\Pi} - \sum_{C_i \text{ is good}} (\|p_i - \hat{p}_i\|) \\
&\geqslant \left(\frac{\sum_{\text{all cubes } C_i} n_i - \sum_{C_i \text{ is bad}} n_i}{\Pi} \right) \\
&- \qquad \frac{\varepsilon^{4d}}{(1024L\sqrt{d})^d} \times N \\
&\geqslant 1 - \left(\frac{\sum_{C_i \text{ is bad}} n_i}{\Pi} \right) - \left(\frac{\varepsilon}{16} \right)^d \\
&\geqslant 1 - \left(\frac{N \times \tau}{\Pi} \right) - \left(\frac{\varepsilon}{16} \right)^d. \\
&\geqslant 1 - \left(\frac{\varepsilon}{16} \right)^d - \left(\frac{\varepsilon}{16} \right)^d.
\end{aligned}$$

Thus,

$$\mathbb{P}_{x\sim\mathbb{V}}[C(x) \text{ is bad}] \leqslant 2\left(\frac{\varepsilon}{16} \right)^d \leqslant \frac{\varepsilon}{8}.$$

Therefore, if $\theta_A(x)$ is the algorithm's output and $\theta^*(x)$ is the optimal threshold, then we get:

$$\begin{aligned}
\mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta_A(x), y)] &= \left(1 + \frac{3\varepsilon}{4} \right) \sum_{C_i \text{ is good}} (\min_{\theta} \mathbb{E}_{x,y\sim\mathbb{D}_i}[g(\theta, y)] \mathbb{P}_{x\sim\mathbb{V}}[x \in C_i]) \\
&+ \qquad 2 \sum_{C_i \text{ is bad}} \mathbb{P}_{x\sim\mathbb{V}}[x \in C_i]) \\
&\leqslant \left(1 + \frac{3\varepsilon}{4} \right) \mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta^*(x), y)] + 2\mathbb{P}_{x\sim\mathbb{V}}[C(x) \text{ is bad}] \\
&\leqslant \left(1 + \frac{3\varepsilon}{4} \right) \mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta^*(x), y)] + \frac{\varepsilon}{4}.
\end{aligned}$$

Since $\mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta^*(x),y)] \geqslant 1$, we have

$$\mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta_A(x),y)] \leqslant \left(1+\frac{3\varepsilon}{4}\right)\mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta^*(x),y)] + \frac{\varepsilon}{4}\cdot\mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta^*(x),y)]$$
$$= (1+\varepsilon)\cdot\mathbb{E}_{(x,y)\sim\mathbb{D}}[g(\theta^*(x),y)].$$

□

2.4.3 Lower Bound Construction

The main shortcoming of Theorem 1 is that there is an exponential dependence of the sample complexity on the number of feature dimensions d. Unfortunately, this dependence is necessary, as shown by the next theorem:

Theorem 3. *For any learning-to-rent algorithm, there exists a family of* 1*-Lipschitz joint distributions* (x,y) *where* $x \in [0,1]^d$ *such that the algorithm must query* $\frac{1}{\varepsilon^{\Omega(d)}}$ *samples in order to be* $(1+\varepsilon)$*-accurate, for small enough* $\varepsilon > 0$*.*

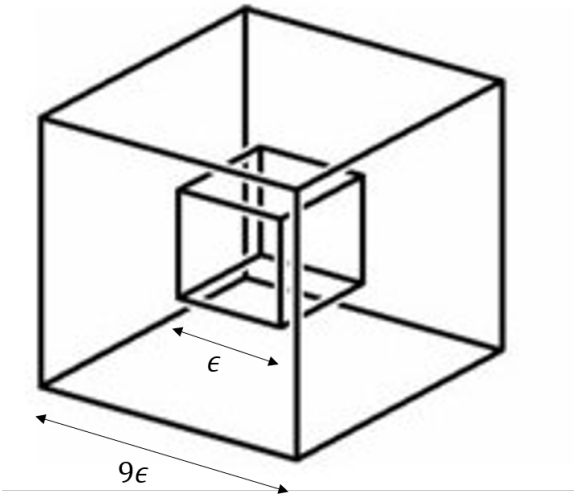

FIGURE 2.2: A sub-Hypercube with the core inside

In the construction, we first divide up the feature space $[0,1]^d$, which is in the form of a hypercube, into smaller hypercubes of side length 9ε. Note that there are $\frac{1}{(9\varepsilon)^d}$ such

sub-hypercubes (see fig 2.2). Next, we define the *core* of each sub-hypercube as a hypercube of side length ε at the center of the sub-hypercube. In other words, the core excludes a boundary of width 4ε in all dimensions. To reduce the effect of the 1-Lipschitz property, we next make two design choices. First, we define x as being uniformly distributed over the cores of all the sub-hypercubes, and the boundary regions have a probability density of 0. Second, the conditional distribution $y|x$ is deterministic and invariant in any core, with value $y = 1-4\varepsilon$ or $y = 1+4\varepsilon$ with probability $1/2$ each.

We now prove two key properties of this family of distributions (x,y). The first lemma shows that we have effectively eliminated the information leakage caused by the 1-Lipschitz property.

Lemma 5. *If an algorithm does not query any sample from a core, then it does not have any information about the conditional distribution $y|x$ in that core.*

Proof. Note that if $x_1, x_2 \in \mathbb{R}^d$ are in different cores, then $\|x_1 - x_2\| \geqslant 8\varepsilon$. This implies that even with the 1-Lipschitz property, the EMD between the conditional distributions $y|x_1$ and $y|x_2$ can be 8ε. Since the two deterministic distributions of $y|x$ used in the construction have this EMD between them, the lemma follows. □

The next lemma establishes that an algorithm that does not have any information about a conditional distribution $y|x$ in any core essentially cannot do better than random guessing.

Lemma 6. *If an algorithm does not query any sample from a core, then its expected competitive ratio on the conditional distribution $y|x$ in that core is at least $1+2\varepsilon$.*

Proof. If a rent-or-buy algorithm is specified only two possible inputs where $y = 1-4\varepsilon$ or $y = 1+4\varepsilon$ (for small enough $\varepsilon > 0$), it has two possible strategies that dominate all others: buy at time 0 or rent throughout. The first strategy achieves a competitive ratio of $\frac{1}{1-4\varepsilon} > 1+4\varepsilon$ for $y = 1-4\varepsilon$ and 1 or $y = 1+4\varepsilon$, whereas the second strategy achieves a competitive ratio of 1 for $y = 1-4\varepsilon$ and $1+4\varepsilon$ or $y = 1+4\varepsilon$. Since the two conditional

distributions are equally likely in a core for the family of joint distributions constructed above, the lemma follows. □

We are now ready to prove Theorem 3

Proof of Theorem 3. Assume, if possible, that the algorithm uses $n = 1/\varepsilon^{d/4}$ samples. Recall that we have $1/(9\varepsilon)^d > 1/\varepsilon^{d/2} = n^2$ sub-hypercubes (for small enough ε) in the construction above. This implies that for at least $1 - 1/n$ fraction of the sub-hypercubes, the algorithm does not get any sample from them. By Lemmas 5 and 6, the competitive ratio of the algorithm on these sub-hypercubes is no better than $1 + 2\varepsilon$. Therefore, even if the algorithm achieved a competitive ratio of 1 on the other sub-hypercubes. the overall competitive ratio is no better than $(1 - 1/n) \cdot (1 + 2\varepsilon) + (1/n) \cdot 1 > 1 + \varepsilon$. The theorem now follows from the observation that an optimal algorithm that knows the conditional distributions $y|x$ in all sub-hypercubes achieves a competitive ratio of 1. □

2.5 PAC Learning Approach

In the previous section, we saw that without making further assumptions, the number of samples required by a learning-to-rent algorithm will be exponential in the dimension of the feature space. To avoid this, we try to identify reasonable assumptions that allow the learning-to-rent algorithm to be more efficient.

We follow the traditional framework of PAC learning. Recall that in PAC learning, the true function mapping features to labels is restricted to a given *concept class* $\mathscr{C}$:

Definition 5. *Consider a set $X \in \mathbb{R}^d$ and a concept class $\mathscr{C}$ of Boolean functions $X \to \{0,1\}$. Let c be an arbitrary hypothesis in $\mathscr{C}$. Let P be a PAC learning algorithm that takes as input the set S comprising m samples (x_i, y_i) where x_i is sampled from a distribution $\mathbb{D}$ on X and $y_i = c(x_i)$, and outputs a hypothesis $\hat{c}$. P is said to be have ε error with failure probability δ, if with probability at least $1 - \delta$:*

$$\mathbb{P}_{x\sim\mathbb{D}}[\hat{c}(x) \neq c(x)] \leqslant \varepsilon.$$

Standard results in learning theory show that if the function class $\mathscr{C}$ is "simple", the PAC-learning problem can be solved with a small number of samples. In the learning-to-rent problem, our goal is to learn the optimal policy $\theta^*(\cdot)$.

We consider the situation where the value of y is deterministic given x. This assumption says that the features contain enough information to predict the length of the ski season.

Assumption 1. *In the input distribution $(x,y) \sim \mathbb{K}$ for the learning-to-rent algorithm, the value of y is a deterministic function of x i.e $y = f(x)$ for some function f.*

Note that in this case, the optimal solution is going to have competitive ratio of 1, so an $(1+\varepsilon)$-accurate learning-to-rent algorithm must have competitive ratio $1+\varepsilon$.

Because of Assumption 1, the entire feature space can be divided into two regions: one where $y < 1$ and renting is optimal, and the other where $y \geqslant 1$ and buying at the outset is optimal. If the boundary between these two regions is PAC-learnable, we can hope to improve on the result from the previous section. This could also be seen as a weakening of Assumption 1:

Assumption 2. *In the input distribution $(x,y) \sim \mathbb{K}$ for the learning-to-rent algorithm where X is the domain for x, there exists a hypothesis $c : X \mapsto \{0,1\}$ lying in a concept class $\mathscr{C}$ such that c separates the regions $y \geqslant 1$ and $y < 1$. For notational purposes, we say $c(x) = 1$ when $y \geqslant 1$ and $c(x) = 0$ when $y < 1$.*

2.5.1 PAC-learning as a black box.

We first show that in this setting, one can use the PAC-learning algorithm as a black-box. In other words, if we can PAC-learn the concept class $\mathscr{C}$ accurately, then we can get a competitive algorithm for the ski-rental problem. The precise algorithm is given in Algorithm 3. Note that we only use Assumption 2 here.

The next theorem relates the competitive ratio achieved by Algorithm 3 with the accuracy of the black-box PAC learner. This implies an upper bound on the sample

Algorithm 3 Black box learning-to-rent algorithm

Set $\tau = \sqrt{\varepsilon}$

Learning: Query n samples. Train a PAC-learner.

For test input x:
if PAC-learner predicts $y \geqslant 1$
then $\theta(x) = \tau$
else $\theta(x) = 1$.

complexity of learning-to-rent, using the sample complexity bounds for PAC learners.

Theorem 4. *Given an algorithm that PAC-learns the concept class $\mathscr{C}$ with error ε and failure probability δ, there exists a learning-to-rent algorithm that has a competitive ratio of $(1+2\sqrt{\varepsilon})$ with probability $1-\delta$.*

Proof. The algorithm first uses PAC-learning as a black box to learn a hypothesis $\hat{c}$. We then set $\theta(x) = 1$ when $\hat{c}(x) = 0$ and setting $\theta(x) = \tau$ (for some small τ that we fix later) when $\hat{c}(x) = 1$.

If $\mathbb{D}$ denotes the distribution of input parameter x then we know that,

$$\mathbb{P}_{x\sim D}[c(x) \neq \hat{c}(x)] \leqslant \varepsilon. \tag{2.8}$$

Obviously, when $\hat{c}(x) = c(x) = 1$, then our worst-case competitive ratio is $1+\tau$. When $\hat{c}(x) = c(x) = 0$, then our competitive ratio is 1. Also with probability ε, $c(x) \neq \hat{c}(x)$ and the worst case competitive ratio is $\max(2, 1+1/\tau)$.

If we use $\tau = \sqrt{\varepsilon}$, we see that the competitive ratio *CR* is bounded above as:

$$\begin{aligned} \text{CR}(\theta, \mathbb{K}) &\leqslant \left(1+\frac{1}{\tau}\right)\cdot\varepsilon + (1-\varepsilon)\cdot(1+\tau) \\ &= 1 + \frac{\varepsilon}{\tau} + \tau\cdot(1-\varepsilon) \leqslant 1 + 2\sqrt{\varepsilon}. \end{aligned}$$

Hence, with probability $1-\delta$, we achieve a competitive ratio of $(1+2\sqrt{\varepsilon})$. The robustness bounds follows immediately from Lemma 8 by noting that $\theta \geqslant \sqrt{\varepsilon}$ for all inputs. □

The above result can be refined for asymmetrical errors (where the classification errors on the two sides are different) showing that the algorithm is more sensitive to errors of one type than the other. Let us first define a PAC learner with asymmetrical errors as follows:

Definition 6. *Given a set $X \in \mathbb{R}^d$ and a concept class C of Boolean functions $X \to \{0,1\}$. Let there be an arbitrary hypothesis $c \in C$. Let P be a PAC learning algorithm that takes as input the set S comprising of m samples (x_i, y_i) where x_i is sampled from a distribution $\mathbb{D}$ on X and $y_i = c(x_i)$, and outputs a hypothesis $\hat{c}$. P is said to be have an (α, β) error with failure probability δ, if with probability at least $1-\delta$ on the sampling of set S.*

$$\mathbb{P}_{x\sim\mathbb{D}}[\hat{c}(x) = 0, c(x) = 1] \leqslant \alpha$$

$$\mathbb{P}_{x\sim\mathbb{D}}[\hat{c}(x) = 1, c(x) = 0] \leqslant \beta$$

We can now show a better bound on the competitive ratio given access an asymmetrical learner.

Theorem 5. *Given an algorithm that PAC learns the concept class C with asymmetrical errors (α, β) and failure probability δ, there exists an algorithm that has a competitive of $(1+3\varepsilon)$ with probability $1-\delta$, where $\varepsilon = \max(\alpha, \sqrt{\beta})$*

Proof. Again we use PAC-learning as a black box to learn a hypothesis $\hat{c}$. We then set $\theta(x) = 1$ when $\hat{c}(x) = 0$ and setting $\theta(x) = \tau$ (for some small τ that will be decided later) when $\hat{c}(x) = 1$

Note that with probability α, $\hat{c}(x) = 0$ and $c(x) = 1$, then we have competitive ratio being capped at 2. And with probability β, $\hat{c}(x) = 1$ and $c(x) = 0$, and our competitive ratio in this case is $\frac{1+\tau}{\tau}$. The rest of the cases, we have the competitive ratio capped at $1+\tau$.

The expected CR is therefore,

$$\begin{aligned}\text{CR} &\leqslant \beta(1+\frac{1}{\tau})+2\alpha+(1-\alpha-\beta)(1+\tau)\\ &= 1+\alpha(1-\tau)+\tau+\frac{\beta}{\tau}\\ &\leqslant 1+\alpha+\tau+\frac{\beta}{\tau}\end{aligned}$$

The CR is minimized at $\tau=\varepsilon=\max(\alpha,\sqrt{\beta})$ and its value is $1+3\varepsilon$. □

Next, we show that the relationship between PAC-learning and learning-to-rent, established in one direction in Theorem 4, actually holds in other direction too. In other words, we can derive a PAC-learning algorithm from a learning-to-rent algorithm. This implies, for instance, that existing lower bounds for PAC-learning also apply to learning-to-rent algorithms. Therefore, in principle, the sample complexity of the algorithm in Theorem 4 is (nearly) optimal without any further assumptions.

Theorem 6. *If there exists an (ε,δ)-accurate learning-to-rent algorithm for a concept class $\mathscr{C}$ with n samples, then there exists an $(4\varepsilon,\delta)$ PAC-learning algorithm for $\mathscr{C}$ with the same number of samples.*

Proof. We will design a PAC learning algorithm (call it P) using the learning-to-rent algorithm (call it A). Given a sample (x_i,z_i) for P, we define sample (x_i,y_i) for A where $y_i=10$ when $z_i=1$, and $y_i=0$ or $y=\frac{1}{2}$ with probability $\frac{1}{2}$ each, when $z_i=0$. The output for P for a feature x is decided as follows: when $\theta(x)\geqslant\frac{1}{2}$ predict $\hat{z}=0$, otherwise, predict $\hat{z}=1$.

First, we calculate the probability $\mathbb{P}[\hat{z}=0,z=1]$. When P predicts $\hat{z}=0$, then we have $\theta(x)\geqslant\frac{1}{2}$. But if, $z=1$, then the optimal cost is 1, whereas the algorithm pays at least $\frac{3}{2}$. Hence the competitive ratio is bounded below by $\frac{3}{2}$. Since the overall competitive ratio is less than $(1+\varepsilon)$, we have that:

$$(1-\mathbb{P}[\hat{z}=0,z=1])+(3/2)\cdot\mathbb{P}[\hat{z}=0,z=1]\leqslant(1+\varepsilon).$$

Therefore, $\mathbb{P}[\hat{z}=0, z=1] \leqslant 2\varepsilon$.

Second, we calculate the error on the other side which is the probability $\mathbb{P}[\hat{z}=1, z=0]$. When P predicts $\hat{z}=1$, then we have $\theta(x) < \frac{1}{2}$. But if $z=0$, then $y=\frac{1}{2}$ with probability $\frac{1}{2}$, and therefore, the competitive ratio is $\geqslant 2$ with probability $\frac{1}{2}$. With the remaining probability of $\frac{1}{2}$, we have $y=0$, and therefore, the competitive ratio is $\geqslant 1$. Therefore, the overall competitive ratio when $\hat{z}=1, z=0$ is $\geqslant \frac{2+1}{2} = \frac{3}{2}$. As earlier, we have $\mathbb{P}[\hat{z}=1, z=0] \leqslant 2\varepsilon$. $\mathbb{P}[\hat{z} \neq z] = \mathbb{P}[\hat{z}=1, z=0] + \mathbb{P}[\hat{z}=0, z=1] \leqslant 4\varepsilon$. □

2.5.2 Margin-based PAC-learning for Learning-to-Rent

Theorem 4 is very general in that there are many concept classes for which we have existing PAC-learning bounds. On the other hand, even for a simple linear separator, PAC-learning requires at least $\Omega(d)$ samples in d dimensions, which can be costly for large d. However, the sample complexity can be reduced when the VC-dimension of the concept class is small:

Theorem 7 (e.g., Kearns and Vazirani (1994))**.** *A concept class of VC-dimension D is (ε, δ) PAC-learnable using $n = \Theta\left(\frac{D+\log(1/\delta)}{\varepsilon}\right)$ samples. For fixed δ, the sample complexity of PAC-learning is $\Theta\left(\frac{D}{\varepsilon}\right)$.*

In particular, this result is used when the underlying data distribution has a *margin*, which is the distance of the closest point to the decision boundary:

Definition 7. *Given a data set $D \in \mathbb{R}^d \times \{0,1\}$ and a separator c, the margin of D with respect to c is defined as* $\min_{x' \in \mathbb{R}^d, (x,y) \in D, c(x') \neq y} \|x - x'\|$.

The advantage of having a large margin is that it reduces the VC-dimension of the concept class. Since the precise dependence of the VC-dimension on the width of the margin (denoted α) depends on the concept class $\mathscr{C}$, let us denote the VC-dimension by $D(\alpha)$.

Crucially, we will show that in the learning-to-rent algorithm, it is possible to *reduce the sample complexity even if the original data* $(x,y) \sim \mathbb{K}$ *does not have any margin*. The main idea is that the learning-to-rent algorithm can ignore training data in a suitably chosen margin. This is because $y \approx 1$ for points in the margin, and the competitive ratio of ski rental is close to 1 for these points even with no additional information. Thus, although the algorithm fails to learn the label of test data near the margin reliably, this does not significantly affect the eventual competitive ratio of the learning-to-rent algorithm.

Note that the L-Lipschitz property under Assumption 1 is:

Assumption 3. *For $x_1, x_2 \in X$ where X is the domain of x, if $y_1 = f(x_1)$ and $y_2 = f(x_2)$, we have $|y_1 - y_2| \leqslant L \cdot \|x_1 - x_2\|$.*

We now give a learning-to-rent algorithm that uses this margin-based approach (Algorithm 4). Recall that α is the width of the margin used by the algorithm; we will set the value of α later.

Algorithm 4 Margin-based learning-to-rent algorithm

Set $\gamma = L\alpha$.

Learning: Query n samples. Discard samples (x_i, y_i) where $y_i \in [1-\gamma, 1+\gamma]$. Use the remaining samples to train a PAC-learner with margin α.

For test input x**:**
if PAC-learner predicts $y \geqslant 1$
then $\theta(x) = \gamma$
else $\theta(x) = 1 + \gamma$.

The filtering process creates an artificial margin:

Lemma 7. *In Algorithm 4, the samples used in the PAC learning algorithm have a margin of α.*

We now analyze the sample complexity of Algorithm 4.

Theorem 8. *Given a concept class $\mathscr{C}$ with VC-dimension $D(\alpha)$ under margin α, there exists a learning-to-rent algorithm that has a competitive ratio of $1+O(L\alpha)$ for n samples with constant failure probability, where α satisfies:*

$$\sqrt{\frac{D(\alpha)}{n}} = L\alpha. \tag{2.9}$$

Proof. Let q denote the probability that (x_i, y_i) satisfies $1-\gamma \leqslant y_i \leqslant 1+\gamma$, i.e., is in the margin. With probability $1-q$, a test input does not lie in the margin and we have the following two scenarios:

- With probability $(1-\varepsilon)$, the prediction is correct and the competitive ratio is at most $(1+\gamma)$.
- With probability ε, the prediction is incorrect and the competitive ratio is at most $\max\left(1+\frac{1}{\gamma}, 2+\gamma\right)$. For small γ (say $\gamma \leqslant 1/2$, which will hold for any reasonable sample size n), this value is $1+\frac{1}{\gamma}$.

With probability q, a test input lies in the margin and the competitive ratio is at most $\frac{1+\gamma}{1-\gamma}$. The expected competitive ratio is:

$$\begin{aligned}
\mathrm{CR}(\theta, \mathbb{K}) &\leqslant (1-q)\cdot(1-\varepsilon)\cdot(1+\gamma) + \\
&\qquad + (1-q)\cdot\varepsilon\cdot\left(1+\frac{1}{\gamma}\right) + q\cdot\left(\frac{1+\gamma}{1-\gamma}\right) \\
&\leqslant 1 + \left[(1-q)\cdot(1-\varepsilon)\cdot\gamma + (1-q)\cdot\varepsilon\cdot\frac{1}{\gamma} + q\cdot\frac{2\gamma}{1-\gamma}\right] \\
&\leqslant 1 + 4\gamma + (1-q)\cdot\frac{\varepsilon}{\gamma} \qquad \text{for } \gamma \leqslant 1/2.
\end{aligned}$$

Now, note that by Chernoff bounds (see, e.g., Motwani and Raghavan (1997)), the number of samples used for training the classifier after filtering is $n_f \geqslant n(1-q)/2$ with constant probability. Also, by Theorem 7 and Lemma 7, we predict whether $y < 1$ or

$y \geqslant 1$ with an error rate of $\varepsilon = O\left(\frac{D(\alpha)}{n_f}\right)$ using n_f samples with constant probability. This implies:

$$(1-q)\cdot\varepsilon = O\left(\frac{D(\alpha)}{n}\right).$$

Thus, $\text{CR}(\theta,\mathbb{K}) \leqslant 1+4\gamma+O\left(\frac{D(\alpha)}{n\cdot\gamma}\right)$. Optimizing for γ, we have $\gamma = \theta\left(\sqrt{\frac{D(\alpha)}{n}}\right)$. But, we also have $\gamma = L\alpha$ in the algorithm. This implies that we choose α to satisfy Eq. (2.9) and obtain a competitive ratio of $1+O(L\alpha)$. □

We now apply this theorem for the important and widely used case of linear separators. The following well-known theorem establishes the VC-dimension of linear separators with a margin.

Theorem 9 (see, e.g., Vapnik and Vapnik (1998))**.** *For an input parameter space $X \in \mathbb{R}^d$ that lies inside a sphere of radius R, the concept class of α-margin separating hyper-planes for X has the VC dimension D given by:*

$$D \leqslant \min\left(\frac{R^2}{\alpha^2}, d\right) + 1.$$

Feature vectors are typically assumed to be normalized to have constant norm, i.e., $R = O(1)$. Thus, Theorem 8 gives the sampling complexity for linear separators as follows:

Corollary 2. *For the class of linear separators, there is a learning-to-rent algorithm that takes as input n samples and has a competitive ratio of* $1+O\left(\frac{\sqrt{L}}{n^{1/4}}\right)$.

For instances where a linear separator does not exist, a popular technique called *kernelization* (see Rasmussen (2003)), is to transform the data points x to a different space $\phi(x)$ where they become linearly separable.

Corollary 3. *For a kernel function ϕ satisfying $\|\phi(x_1)-\phi(x_2)\| \geqslant \frac{1}{\nu}\cdot\|x_1-x_2\|$ for all x_1, x_2, assuming the data is linearly separable in kernel space, there exists a learning-to-rent algorithm that achieves a competitive ratio of $1+O\left(\frac{\sqrt{L\nu}}{n^{1/4}}\right)$ with n samples,*

Conceptually, the corollary states that we can make use of these kernel mappings without hurting the competitive ratio bounds achieved by the algorithm. This is because the sample complexity in the margin-based algorithm (Algorithm 4) is independent of the number of dimensions.

2.6 Learning-to-rent with a Noisy Classifier

So far, we have seen that PAC-learning a binary classifier with deterministic labels (Assumption 1) is sufficient for a learning-to-rent algorithm. However, in practice, the data is often noisy, which leads us to relax Assumption 1 in this section. Instead of requiring $y|x$ to be deterministic, we only insist that $y|x$ is predictable with sufficient probability. In other words, we replace Assumption 1 with the following (weaker) assumption:

Assumption 1'. *In the input distribution $(x,y) \sim \mathbb{K}$, there exists a deterministic function f and a parameter p such that the conditional distribution of $y|x$ satisfies $y = f(x)$ with probability at least $1-p$.*

This definition follows the setting of binary classification with noise first introduced by Bylander (1994). Indeed, the existence of noise-tolerant binary classifiers (e.g., Blum et al. (1998); Awasthi et al. (2014); Natarajan et al. (2013)), leads us to ask if these classifiers can be utilized to design learning-to-rent algorithms under Assumption 1'. We answer this question in the affirmative by designing a learning-to-rent algorithm in this noisy setting (see Algorithm 5). This algorithm assumes the existence of a binary classifier than can tolerate a noise rate of p and achieves classification error of ε. Let $p_0 = \max(p, \varepsilon)$. If p_0 is large, then the noise/error rate is too high for the classifier to give reliable information about test data; in this case, the algorithm reverts to a worst-case (randomized) strategy.

On the other hand, if p_0 is small, the the algorithm uses the label output by the classifier, but with a minimum wait time of $\sqrt{p_0}$ on all instances to make it robust to noise and/or classification error.

Algorithm 5 Learning-to-rent with a noisy classifier

Set $p_0 = \max(p, \varepsilon)$.

Learning:
if $p_0 \leqslant \frac{1}{9(e-1)^2}$
then PAC-learn the classifier on n (noisy) training samples.

For test input x:
if $p_0 > \frac{1}{9(e-1)^2}$
then $\mathbb{P}[\theta(x) = z] = \begin{cases} \frac{e^z}{e-1}, \text{ for } z \in [0,1] \\ 0, \text{ for } z > 1. \end{cases}$
else
  if PAC-learner predicts $y < 1$
  then $\theta(x) = 1$
  else $\theta(x) = \sqrt{p_0}$.

The next theorem shows that this algorithm has a competitive ratio of $1 + O(\sqrt{p_0})$ for small p_0, and does no worse than the worst case bound of $\frac{e}{e-1}$ irrespective of the noise/error:

Theorem 10. *If there is a PAC-learning algorithm that can tolerate noise of p and achieve accuracy ε, the above algorithm achieves a competitive ratio of* $\min(1 + 3\sqrt{p_0}, \frac{e}{e-1})$ *where* $p_0 = \max\{p, \varepsilon\}$.

Proof of Theorem 10. Note that if $p_0 > \frac{1}{9(e-1)^2}$, we choose the threshold θ according to:

$$\Pr[\theta = z] = \begin{cases} \frac{e^z}{e-1}, \text{ for } z \in [0,1] \\ 0, \text{ for } z > 1. \end{cases}$$

It can be verified by taking the expectation that $\mathbb{E}[Alg] = \frac{e}{e-1} \times \min\{y, 1\}$ and we obtain the competitive ratio $\frac{e}{e-1}$ Karlin et al. (1994). We now assume that $p_0 < \frac{1}{9(e-1)^2}$ for the rest of the proof.

We first focus on the points where the PAC learner's prediction is correct. This is indeed true for $1-\varepsilon$ fraction of the samples from the distribution, where the expectation is taken over the probability distribution of the samples.

If $y > 1$, then the algorithm chooses to buy at $\sqrt{p_0}$, the adversary can flip the label and cause the CR (in the worst-case) to become $1 + \frac{1}{\sqrt{p_0}}$ (this happens with probability p), and otherwise, the competitive ratio is upper bounded by $(1+\sqrt{p_0})$ (this occurs with probability $\leqslant 1-p$). Hence, in expectation the competitive ratio is therefore $p\left(1+\frac{1}{\sqrt{p_0}}\right) + (1-p)(1+\sqrt{p_0}) < 1+\sqrt{p_0}+\frac{p}{\sqrt{p_0}}$.

When $y < 1$ and $p \leqslant p_0 \leqslant \frac{1}{9(e-1)^2}$, we buy at 1 and our competitive ratio is 2 with probability p (adversarial) and 1 with probability $1-p$ (no adversary). Hence, the expected competitive ratio is $1+2p$. The competitive ratio when the PAC learner is correct is therefore, $\max\{1+2p, 1+\sqrt{p_0}+\frac{p}{\sqrt{p_0}}\} \leqslant (1+2\sqrt{p_0})$

Now we focus on the points on which the PAC learner makes an error. These comprise ε fraction of the points in the distribution. When y is predicted to be $\leqslant 1$ and is actually > 1, then our competitive ratio is upper bounded by 2 (since we our always buying before y exceeds 1 and the optimal solution pays 1). When y is predicted to be > 1 but is actually $y < 1$, then the worst case competitive ratio is $1+\frac{1}{\sqrt{p_0}}$.

We are now ready to calculate the expected competitive ratio as follows:

$$\begin{aligned}
\text{CR} &\leqslant (1+2\sqrt{p_0})\cdot(1-\varepsilon) + \varepsilon\cdot\left(1+\frac{1}{\sqrt{p_0}}\right) \\
&\leqslant 1+2(1-\varepsilon)\sqrt{p_0} + \left(\frac{\varepsilon}{\sqrt{p_0}}\right) \\
&\leqslant 1+3\sqrt{p_0}.
\end{aligned}$$

□

We also show that the above result is optimal in a rather strong sense: namely, even with no classification error, the competitive ratio achieved cannot be improved.

Theorem 11. *For a given noise rate $p \leqslant \frac{1}{2}$, no (randomized) algorithm can achieve a competitive ratio smaller than $1 + \frac{\sqrt{p}}{2}$, even when the algorithm has access to a PAC-learner that has zero classification error.*

Proof. We will show that the adversary can choose a distribution on supplying y that yields a large competitive ratio regardless of the θ that the algorithm chooses. Let's focus when $y > 1$ and the PAC learner correctly predicts this surely.

If there was no adversary, the algorithm should buy at 0 and CR is 1. However the presence of an adversary makes it a bad move, since the adversary can pick $y = \rho$ for an arbitrarily small but positive ρ with a non-zero probability and drive up the competitive ratio arbitrarily.

Here is the exact strategy that the adversary chooses to hurt the algorithm: the distribution on y is $g(y) = kye^{-y}$. for $y \in [0, \sqrt{p}]$ (k being the normalization constant) This is quite similar to the adversarial distribution chosen in Karlin et al. (1994) to enforce an $\frac{e}{e-1}$ ratio.

Now for any value $\theta \in (0, \sqrt{p})$ that the algorithm chooses, the competitive ratio is given by:

$$\text{CR}(\theta, \mathbb{K}) = p \cdot \left[\int_0^{\theta} g(y)dy + \int_{\theta}^{\sqrt{p}} \frac{(1+\theta)}{y} g(y)dy \right] + (1+\theta)(1-p).$$

Calculating the derivative with respect to θ, we get:

$$\begin{aligned}
\frac{d(\text{CR}(\theta, \mathbb{K}))}{d(\theta)} &= pg(\theta) + p \int_{\theta}^{\sqrt{p}} \frac{g(y)}{y} dy - p\frac{(1+\theta)}{\theta} g(\theta) + (1-p) \\
&= p\frac{g(\theta)}{\theta} + p \int_{\theta}^{\sqrt{p}} ke^{-y} dy + (1-p) \\
&= -pke^{-\theta} + pk(e^{-\theta} - e^{-\sqrt{p}}) + (1-p) \\
&= -pke^{-\sqrt{p}} + (1-p)
\end{aligned}$$

Using the fact that the total probability $\int_0^{\sqrt{p}} g(y)dy = 1$ we get that $k = \frac{1}{(1-(1+\sqrt{p})e^{-\sqrt{p}})}$.

It is easy to see that this value of k gives: $\frac{d(\text{CR}(\theta, \mathbb{K})}{d(\theta)} \leqslant 0$.

Hence $\text{CR}(\theta, \mathbb{K})$ decreases as θ goes from 0 to $\sqrt{p}$. Also, the algorithm gains nothing by increasing θ beyond $\sqrt{p}$. Hence, the best competitive ratio is obtained when algorithm chooses $\theta = \sqrt{p}$. Thus, the algorithm can't hope for a competitive ratio better than

$$\begin{aligned}\text{CR}(\sqrt{p}, \mathbb{K}) &= p\int_0^{\sqrt{p}} g(y)dy + (1+\sqrt{p})(1-p) \\ &= p + (1+\sqrt{p})(1-p) \\ &= 1 + \sqrt{p} - p\sqrt{p}\end{aligned}$$

For $p < 1/2$:

$$\geqslant 1 + \frac{\sqrt{p}}{2}$$

□

2.7 Robustness Bounds

In this section, we address the scenario when there is no assumption on the input, i.e., the choice of the input is adversarial. The desirable property in this setting is encapsulated in the following definition of "robustness" adapted from the corresponding notion in Purohit et al. (2018):

Definition 8. *A learning-to-rent algorithm A with threshold function $\theta(\cdot)$ is said to be γ-robust if $g(\theta(x), y) \leqslant \gamma$ for any feature x and any length of the ski season y.*

First, we show an upper bound on the competitive ratio for any algorithm based on the shortest wait time for any input.

Lemma 8. *A learning-to-rent algorithm with threshold function $\theta(\cdot)$ is $\left(1 + \frac{1}{\theta_0}\right)$-robust where:*

$$\theta_0 = \min_{x \in \mathbb{R}^d} \theta(x).$$

Proof. Note that the function $g(\theta, y)$ achieves its maximum value at $y = \theta + \rho$ where $\rho \to 0^+$. In this case, the algorithm pays $1 + \theta$, while the optimal offline cost approaches θ. This gives us that $\max_{y \in \mathbb{R}^+} g(\theta, y) = \left(1 + \frac{1}{\theta}\right)$. Now, since there is no x such that $\theta(x) < \theta_0$, we get:

$$\max_{y \in \mathbb{R}^+, x \in \mathbb{R}^d} g(\theta(x), y) \leqslant \left(1 + \frac{1}{\theta_0}\right)$$

. □

The robustness bounds for our algorithms are straightforward applications of the above lemma. We derive these bounds below. First, we consider Algorithm 2 based only on the Lipschitz assumption.

Theorem 12. *Algorithm 2 is* $\left(1 + \frac{1}{\varepsilon}\right)$*-robust.*

Proof. Algorithm 2 always chooses a threshold in the range $[\varepsilon, 1/\varepsilon]$, i.e., $\theta \geqslant \varepsilon$ for all inputs. The theorem now follows by Lemma 8. □

Next, we consider the black box algorithm that uses the PAC learning approach, i.e., Algorithm 3.

Theorem 13. *Algorithm 3 is* $\left(1 + \frac{1}{\sqrt{\varepsilon}}\right)$*-robust.*

Proof. Note that Algorithm 3 has $\theta \geqslant \sqrt{\varepsilon}$ for all inputs, which by Lemma 8 gives a robustness bound of $1 + \frac{1}{\sqrt{\varepsilon}}$. □

Next, we show robustness bounds for the margin-based approach, i.e., Algorithm 4.

Theorem 14. *Algorithm 4 is* $\left(1 + \frac{1}{L\alpha}\right)$*-robust.*

Proof. This follows from Lemma 8, with the observation that the shortest wait time in Algorithm 4 is $\gamma = L\alpha$. □

Finally, we consider the noisy classification setting in Algorithm 5.

Theorem 15. *Algorithm 5 is* $\max\left(\frac{e}{e-1}, 1+\frac{1}{\sqrt{\varepsilon}}\right)$*-robust.*

Proof. In the two cases in Algorithm 5, either the threshold θ satisfies $\theta \geqslant \sqrt{p_0}$ or a random threshold is chosen for which the expected competitive ratio is $\frac{e}{e-1}$ for any input. In the first, case, we further note that $p_0 = \max(p, \varepsilon) \geqslant \varepsilon$, i.e., $1+\frac{1}{\sqrt{p_0}} \leqslant 1+\frac{1}{\sqrt{\varepsilon}}$. The theorem now follows by applying Lemma 8. □

2.8 Experimental Simulations

In this section, we use numerical simulations to evaluate the algorithms that we designed for the learning-to-rent problem: the black box algorithm (Algorithm 3), the margin-based algorithm (Algorithm 4), and the algorithm for a noisy classifier (Algorithm 5). We compare the first two algorithms and show that as the predicted by the theoretical analysis, the margin-based algorithm substantially outperforms the black box algorithm in high dimensions. For learning-to-rent with a noisy classifier, we show that its competitive ratio follows the $(1+\sqrt{p})$-curve predicted by the theoretical analysis with increasing noise rate p.

Experimental Setup. We first describe the joint distribution $(x, y) \sim \mathbb{D}$ used in the experiments. We choose a random vector $W \in \mathbb{R}^d$ as $W \sim N(0, \mathbf{I}/d)$. We view W as a hyper-plane passing through the origin ($W^T x = 0$). The value of y, representing the length of the ski season, is calculated as $\frac{2}{(1+e^{-W^T x})}$, such that $y \geqslant 1$ when $W^T x \geqslant 0$ and $y < 1$ otherwise. Note that this satisfies the Lipschitz condition given in Definition 4, with $L = 2$ for $\|W\| \leqslant 1$. The input x is drawn from a mixture distribution, where with probability $1/2$ we sample x from a Gaussian $x \sim N(0, \mathbf{I}/d)$, and with probability $1/2$, we sample x as $x = \alpha W + \eta$, here $\alpha \sim N(0,1)$ is a coefficient in the direction of W and $\eta \sim N(0, \frac{1}{d}I)$. Choosing x from the Gaussian distribution ensures that the data-set has no margin; however, in high dimensions, $W^T x$ will concentrate in a small region, which makes all the label y very close to 1. We address this issue by mixing in the second component which ensures

that the distribution of y is diverse.

Training and Validation. For a given training set, we split it in two equal halves, the first half is used to train our PAC learner and the second half is used as a validation set to optimize the design parameters in the algorithms, namely τ in Algorithm 3 and γ in Algorithm 4.

Parameter Optimization for Algorithm 3 and Algorithm 4. We perform this optimization on a validation set that is distinct from the training set for these algorithms.

For the black box algorithm (Algorithm 3), we have to choose the value of the parameter τ. Here we set $\tau = c\varepsilon$ where $c > 0$ is a parameter that we optimize on the validation set. In order to do this, we minimize the loss (in this case, the competitive ratio) by running gradient descent from a starting value c_0, where $c_0 \in \{1000, 100, 10, 1, 0.1, 0.01\}$.

For the margin based learning-to-rent algorithm (Algorithm 4), we optimize the value of γ using a similar procedure by running gradient descent from the starting value $\frac{\gamma_0}{N^{1/4}}$, where $\gamma_0 \in \{0.1, 0.01, 0.001, 0.0001, 10^{-5}\}$.

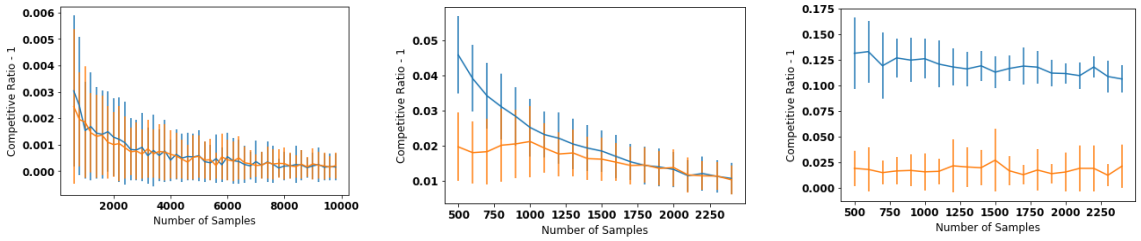

FIGURE 2.3: Comparison of Algorithm 3 (blue) and Algorithm 4 (orange). From left to right, $d = 2, 100$, and 5000.

We test our algorithms for dimensions $d = 2, 100$, and 5000. For each d, we create a large corpus of samples and select N of them randomly and designate this as the training set; the remaining samples form the test set.

Comparison between the two algorithms. The comparative performance of Algorithm 3 and Algorithm 4 for $d = 2, 100$, and 5000 is given in Fig. 2.3.[4] For small d ($d = 2$),

[4] In all the figures, the vertical bars represent standard deviation of the output value and the value plotted on

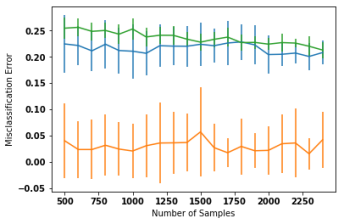

FIGURE 2.4: Classification error in Algorithm 3 (green) and Algorithm 4 (blue for all samples, orange for filtered samples).

we do not see a significant difference in the performance of the two algorithms because the curse of dimensionality suffered by Algorithm 3 is not prominent at this stage. In fact, in this case the optimal margin on validation set is very close to 0. However, as d increases, Algorithm 4 starts outperforming Algorithm 3 as expected from the theoretical analysis. For $d = 100$, this difference of performance is prominent at small sample size but disappears for larger samples, because of the trade-off between sample size and number of dimensions in Corollary 2 and Theorem 4. Eventually, at $d = 5000$, Algorithm 4 is clearly superior.

To further understand the difference between the black box approach and the margin-based approach, in Figure 2.4, we plot the error of the two binary classifiers used in Algorithm 3 and Algorithm 4 when $d = 5000$. Although both classifiers achieve very low accuracy on the entire data-set, the margin-based classifier was able to correctly label the data points that are far from the decision boundary, i.e., the data points where misclassification would be costly from the optimization perspective. As a result, Algorithm 4 performs much better overall.

Learning with noise. We now evaluate the learning-to-rent algorithm with a noisy classifier (Algorithm 5), We fix the number of dimensions $d = 100$, and create a training set of $N = 10^5$ samples using the same distribution as earlier. But now, we add noise to the data by declaring each data point as noisy with probability p (we will vary the parameter p over our experiments). There are two types of noisy data points: ones where the classifier predicts $y \geqslant 1$ and the actual value is $y < 1$, or vice-versa. For data points of the first type,

the curve is the mean.

we choose y from the worst case input distribution in the lower bound given by Theorem 11, i.e, $\mathbb{P}[y = z] = \frac{e}{e-1} \cdot z \cdot e^{-z}$ for $z \in [0,1]$ and point mass of $1/(e-1)$ at some $z > 1$, say at $z = 2$. For data points of the second type, the input distribution is not crucial, so we simply choose a uniform random y in $[1,2]$. The testing is done on a batch of 1000 samples from the same distribution. We use a noise tolerant Perceptron Learner (see, e.g., Bylander (1994)) to learn the classes ($y \geqslant 1$ and $y < 1$) in the presence of noise. We can see that even for noise rates as high as 40%, the competitive ratio of the learning-to-rent algorithm is still better than the $\frac{e}{e-1}$ that is the best achievable in the worst case. (Figure 2.5)

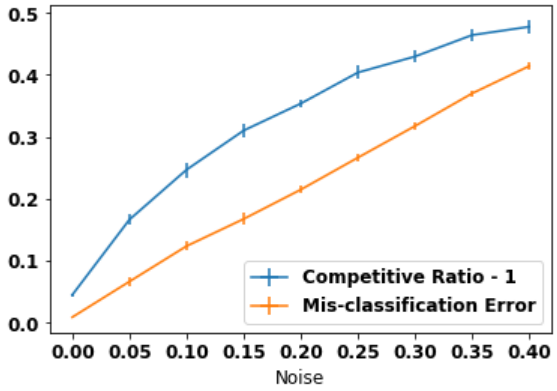

FIGURE 2.5: Algorithm 5 with varying noise rate with $d = 100$.

2.9 Conclusion

In this chapter, we explored the question of customizing machine learning algorithms for optimization tasks, by incorporating optimization objectives in the loss function. We demonstrated, using PAC learning, that for the classical rent or buy problem,the sample complexity of learning can be substantially improved by incorporating the insensitivity of the objective to mis-classification near the classification boundary (which is responsible for large sample complexity if accurate classification were the end goal). In addition, we showed worst-case robustness bounds for our algorithms, i.e., that they exhibit bounded competitive ratios even if the input is adversarial.

2.10 Chapter Notes

The work presented in this chapter was done jointly with Rong Ge and Debmalya Panigrahi. It appeared in the Proceedings of the 37th International Conference on Machine Learning, PMLR 119:303-313 (see Anand et al. (2020))

3

A Regression Approach for ONLINESEARCH

3.1 Introduction

As seen in the last chapter, supplementing an algorithm with reasonably accurate predictions can vastly improve its performance. In the previous chapter, we relied on a classification learning approach for the Ski Rental problem, where we aimed to learn a function that maps the feature set to a binary label characterizing the optimal solution. However, in general, the value of the optimal solution is a real-valued function, which motivates a regression approach to learning-augmented online algorithms that we elucidate in this chapter.

3.1.1 Background

At the time of writing this thesis, there has been very recent research that focuses on the learnability of predicted parameters in online algorithms. Recently, Lavastida *et al.* Lavastida et al. (2020), building on the work of Lattanzi *et al.* Lattanzi et al. (2020), took a data-driven algorithms approach to design online algorithms for scheduling and matching problems via *learned weights*. In this line of work, the goal is to observe sample inputs in order to learn a set of weights that facilitate better algorithms for instances from a fixed distribution. Anand *et al.* Anand et al. (2020) relied on a classification learning approach

for the Ski Rental problem.

3.1.2 Problem Definition

To formalize the notion of an unknown optimal solution that we seek to learn via regression, we use the *online search* (ONLINESEARCH) framework. In this framework, there is as an input sequence $\Sigma = \sigma_1, \sigma_2, \ldots$ available offline, and the actual online input is a prefix of this sequence $\Sigma_T = \sigma_1, \sigma_2, \ldots, \sigma_T$, where the length of the prefix T is revealed online. Namely, in each online step $t > 0$, there are two possibilities: either the sequence *ends*, i.e., $T = t$, or the sequence *continues*, i.e., $T > t$. The algorithm must maintain, at all times t, a solution that is feasible for the current sequence, i.e., for the prefix $\Sigma_t = \sigma_1, \ldots, \sigma_t$. The goal is to obtain a solution that is of minimum cost among all the feasible solutions for the actual input sequence Σ_T.

We make some mild assumptions on the problem.

- *Monotonicity:* Any feasible solution for an input sequence is also feasible for any prefix of the sequence. Moreover, the cost of the optimal solution is monotonically non-decreasing in the length of the prefix.

- *Composibility:* If we add a feasible solution for a input sequence to an existing solution, the overall solution remains feasible for the sequence.

- *Solvability:* The offline problem is solvable, i.e., given a prefix of the input sequence, there is an algorithm that outputs the optimal solution for the prefix.

These assumptions hold for essentially all online problems we care for.

3.1.3 Our Contributions

We will give a sketch of our main approach and results in this chapter. We will first design a simple strategy for the ONLINESEARCH problem without predictions and then explore how predictions can aid such an algorithm. Then we explore the design of the prediction

pipeline; show the inadequacy of traditional loss metric and instead motivate the design of a cognizant loss function that is synergistic with the performance measure of the online algorithm.

Our Approach

As a warm up, we first give an algorithm called DOUBLE for the ONLINESEARCH problem *without predictions* in Section 3.4. The DOUBLE algorithm has a competitive ratio of 4. We build on the DOUBLE algorithm in Section 3.5, where we give an algorithm called PREDICT-AND-DOUBLE for the ONLINESEARCH problem *with predictions*. We show that the PREDICT-AND-DOUBLE algorithm has a consistency of $1+\varepsilon$ and robustness of $O(1/\varepsilon)$, for any hyper-parameter $\varepsilon > 0$. We also show that this tradeoff between consistency and robustness is asymptotically tight.

Our main contributions are in Section 3.6. In this section, we model the question of obtaining a learning-augmented algorithm for the ONLINESEARCH problem in a regression framework. Specifically, we assume that the input comprises a feature vector x that is mapped by an unknown real-valued function f to an input for the ONLINESEARCH problem z. In the training phase, we are given a set of labeled samples of the form (x,z) from some (unknown to the algorithm) data distribution $\mathbb{D}$. The goal of the learning algorithm is to produce a mapping from the feature space to algorithmic strategies for the ONLINESEARCH problem, such that when it gets an unlabeled (test) sample x from the same distribution $\mathbb{D}$, the algorithmic strategy corresponding to x obtains a competitive solution for the actual input z in the test sample (that is unknown to the algorithm).

The learning algorithm employs a regression approach in the following manner. It assumes that the function f is from a hypothesis class $\mathscr{F}$, and obtains an empirical minimizer in $\mathscr{F}$ for a carefully crafted loss function on the training samples. The design of this loss function is crucial since a bound on this loss function is then shown to translate to a bound on the competitive ratio of the algorithmic strategy. (Indeed, we will show later that because

of this reason, standard loss functions used in regression are inadequate for our purpose.) Finally, we use statistical learning theory for real-valued functions to bound the sample complexity of the learner that we design.

Using the above framework, we show a sample complexity bound of $O\left(\frac{H \cdot d}{\varepsilon}\right)$ for obtaining a competitive ratio of $1+\varepsilon$, where H and d respectively represent the log-range of the optimal cost and a measure of the expressiveness of the function class $\mathscr{F}$ called its pseudo-dimension.[1] We also extend this result to the so-called agnostic setting, where the function class $\mathscr{F}$ is no longer guaranteed to contain an exact function f that maps x to z, rather the competitive ratio is now in terms of the *best* function in this class that approximates f. We also prove nearly matching lower bounds for our sample complexity bounds in the two models.

Our framework can also be extended to the setting where the offline optimal solution is hard to compute, but there exists an algorithm with competitive ratio c given the *cost* of optimal solution. In that case our algorithms gives a competitive ratio $c(1+\varepsilon)$, which can still be better than the competitive ratio without predictions (see examples in next subsection).

3.2 Applications of the ONLINESEARCH framework

The ONLINESEARCH framework is applicable *whenever an online algorithm benefits from knowing the optimal value of the solution*. Many online problems benefit from this knowledge, which is sometimes called *advice* in the online algorithms literature. For concreteness, we give three examples of classic problems – *ski rental with multiple options*, *online scheduling*, and *online bin packing* – to illustrate the applicability of our framework. Our algorithm PREDICT-AND-DOUBLE (explained in more detail in section 3.5) successively predicts the optimal value of the solution and appends the corresponding solution to its

[1] Intuitively, the notion of pseudo-dimension extends that of the well-known VC dimension from binary to real-valued functions.

output.

Ski Rental with Multiple Options. Generalizations of the ski rental problem with multiple options have been widely studied (e.g., Ai et al. (2014); Lotker et al. (2012); Meyerson (2005b); Fleischer (2001)), recently with ML predictions Wang et al. (2020). Suppose there are V options (say coupons) at our disposal, where coupon i costs us C_i and is valid for d_i number of days. Given such a setup, we need to come up with a schedule: $\{(t_k, i_k), k = 1, 2 \ldots\}$ that instructs us to buy coupon i_k at time t_k. (The classic ski rental problem corresponds to having only two coupons $C_1 = 1, d_1 = 1$ and $C_2 = B, d_2 \to \infty$.) Our ONLINESEARCH framework is applicable here: a solution that allows us to buy coupons valid time t is also a valid solution for all times $s \leqslant t$. Further, PREDICT-AND-DOUBLE can be implemented efficiently as we can compute $\text{OPT}(t)$, for any time t using a dynamic program.

Online Scheduling. Next, we consider the classic online scheduling problem where the goal is to assign jobs arriving online to a set of identical machines so as to minimize the maximum load on any machine (called the *makespan*). For this algorithm, the classic list scheduling algorithm Graham (1969) has a competitive ratio of 2. A series of works Galambos and Woeginger (1993); Bartal et al. (1992); Karger et al. (1996); Albers (1999) improved the competitive ratio to 1.924, and currently the best known result has competitive ratio of (approx) 1.92 Fleischer and Wahl (2000); in fact, there are nearly matching lower bounds Gormley et al. (2000). However, if the optimal makespan (OPT) is *known*, then these lower bounds can be overcome, and a significantly better competitive ratio of 1.5 can be obtained in this setting Böhm et al. (2017) (see also Azar and Regev (1998); Kellerer and Kotov (2013); Gabay et al. (2015, 2017)). The ONLINESEARCH framework is applicable here with a slight modification: whenever PREDICT-AND-DOUBLE tries to buy a solution corresponding to a predicted value of OPT, we execute the 1.5-approximation algorithm based on this value. The problem still satisfies the property that a solution for t jobs is valid for any prefix. We get a competitive ratio of $1.5 + O(\varepsilon)$ that significantly outperforms the

competitive ratio of 1.92 without predictions.

Online Bin Packing. As a third example, we consider the online bin packing problem. In this problem, items arrive online and must be packed into fixed-sized bins, the goal being to minimize the number of bins. (We can assume w.l.o.g., by scaling, that the bins are of unit size.) Here, it is known that the critical parameter that one needs to know/predict is not OPT but the number of items of *moderate* size, namely those sized between $1/2$ and $2/3$. If this is known, then there is a simple 1.5-competitive algorithm Angelopoulos et al. (2015), which is not achievable without this additional knowledge. Again, our ONLINESEARCH framework can be used to take advantage of this result. In this case, the application is not as direct, because predicting OPT does not yield the better algorithm. Nevertheless, an inspection of the algorithm in Angelopoulos et al. (2015) reveals the following strategy: The items are partitioned into three groups. The items of size $\geqslant 2/3$ are assigned individual bins, items of size between $1/3$ and $1/2$ are assigned separate bins where at least two of them are assigned to each bin, and the remaining items are assigned a set of common bins. Clearly, the first two categories can be handled online without any additional information; this means that we can define a surrogate OPT (call it OPT') that only captures the optimal number of bins for the common category. Note that the of prediction of OPT' serves as a substitute for knowing the numbers of items of moderate size. Now, if OPT' is known, then we can recover the competitive ratio of $3/2$ by using a simple greedy strategy. This now allows us to use the ONLINESEARCH framework where we predict OPT'. As earlier, the ONLINESEARCH framework can be applied with slight modification: whenever PREDICT-AND-DOUBLE tries to buy a solution corresponding to a predicted value of OPT', we execute the 1.5-competitive algorithm based on this value. The problem still satisfies the property that a solution for t items is valid for any prefix.

3.3 LEARNTOSEARCH

An instance (x,z) of the LEARNTOSEARCH problem is given by a feature $x \in \mathbb{X}$, and the (unknown) cost of the optimal offline solution $z \in [1,M]$. The two quantities x and z are assumed to be drawn from a joint distribution on $\mathbb{X} \times [1,M]$. A prediction strategy works with a hypothesis class $\mathscr{F}$ that is a subset of functions $\mathbb{X} \mapsto [1,M]$ and tries to obtain the best function $f \in \mathscr{F}$ that predicts the target variable z accurately. For notational convenience, we set our target $y = \ln z$, i.e., we try to predict the log-cost of the optimal solution. Note that predicting the log-cost of $\text{OPT}(T)$ is equivalent to predicting the input length T.[2] Furthermore, let $\mathbb{D}$ denote the input distribution on $\mathbb{X} \times \mathbb{Y}$, where $\mathbb{Y} = [0,H]$ and $H = \ln M$; i.e., we assume that $(x,y) \sim \mathbb{D}$.

We define a LEARNTOSEARCH algorithm $\mathscr{A}$ as a strategy that receives a set of m samples $S \sim \mathbb{D}^m$ for *training*, and later, when given the feature set x of a *test* instance $(x,y) \sim \mathbb{D}$ (where y is not revealed to the algorithm), it defines an online algorithm for the ONLINESEARCH problem with input y. Recall that an online algorithm constitutes a sequence of solutions that the algorithm buys at different times of the input sequence.

Definition 9. *We use* $\text{OPT}(t)$ *to denote an optimal (offline) solution for the input prefix of length* t*; we overload notation to denote the cost of this solution by* $\text{OPT}(t)$ *as well.*

We will use the notation $\text{CR}_{\mathscr{A}}(x,y)$ to denote the competitive ratio obtained by an algorithm $\mathscr{A}$ on the instance (x,y). For a given set of thresholds $(\tau_0, \tau_1 \ldots)$, define $i_T = \min_{\tau_i > T} i$. Then, $\mathscr{A}$ pays a total cost of $\sum_{i=0}^{i_T} \text{OPT}(\tau_i)$, and thus the competitive ratio is

$$\text{CR}_{\mathscr{A}}(x,y) = \frac{\sum_{i=0}^{i_T} \text{OPT}(\tau_i)}{e^y}.$$

We define the "efficiency" of a LEARNTOSEARCH algorithm by comparing its performance with the best achievable competitive ratio. The optimal competitive ratio for a given

[2] When multiple input lengths might have the same optimal cost, we can just pick the longest one.

distribution may be strictly greater than 1. For example, consider the distribution where x is fixed (say x_0) and z is uniformly distributed over the set $\{2,4\}$. One can verify that the best strategy for the above distribution is to buy the solution of cost 2, and then if the input has not ended, then buy the solution of cost 4. The competitive ratio for this strategy (in expectation) is 1.25.

Definition 10. *A* LEARNTOSEARCH *algorithm $\mathscr{A}$ is said to be ε-efficient if*

$$\mathbb{E}_{(x,y)\sim\mathbb{D}}\mathrm{CR}_{\mathscr{A}}(x,y) \leqslant \rho^* + \varepsilon,$$

where $\rho^* = \mathbb{E}_{(x,y)\sim\mathbb{D}}\mathrm{CR}_{\mathscr{A}^*}(x,y)$ and $\mathscr{A}^*$ is an optimal solution that has full knowledge of $\mathbb{D}$ and no computational limitations.

The "expressiveness" of a function family is captured by the following standard definition:

Definition 11. *A set $S = \{x_1, x_2, \ldots x_m\}$ is said to be "shattered" by a class $\mathscr{F}$ of real-valued functions $S \mapsto [0,H]$ if there exists "witnesses" $R = \{r_1, r_2 \ldots r_m\} \in [0,H]^m$ such that the following condition holds: For all subsets $T \subseteq S$, there exists an $f \in \mathscr{F}$ such that $f(x_i) > r_i$ if and only if $x_i \in T$. The "pseudo-dimension" of $\mathscr{F}$ (denoted as $Pdim(\mathscr{F})$) is the cardinality of the largest subset $S \subseteq X$ that is shattered by $\mathscr{F}$.*

3.4 ONLINESEARCH without Predictions

As a warm up, we first describe a simple algorithm called DOUBLE (Algorithm 6) for the ONLINESEARCH problem *without predictions.* This algorithm places milestones on the input sequence corresponding to inputs at which the cost of the optimal solution doubles. When the input sequence crosses such a milestone, the algorithm buys the corresponding optimal solution and adds it to the existing online solution. This simple algorithm will form a building block for the algorithms that we will develop later in the chapter; hence, we describe it and prove its properties below.

Definition 12. *Given an input length τ and any $\alpha > 0$, we use* MIN-LENGTH(α, τ) *to denote the smallest length t such that* OPT$(t) \geqslant \alpha \cdot$ OPT(τ). *The monotonicity property of* OPT *ensures that* MIN-LENGTH$(\alpha, \tau) > \tau$ *if $\alpha > 1$, and* MIN-LENGTH$(\alpha, \tau) \leqslant \tau$ *otherwise.*

Algorithm 6 DOUBLE

Input: The input sequence $\mathscr{I}$.
Output: The online solution SOL.

Set $i := 0$, $\tau_0 := 1$, SOL $:= \varnothing$.
for $t = 1, 2, \ldots, T$
  if $t = \tau_i$
    Set $\tau_{i+1} =$ MIN-LENGTH$(2, \tau_i)$.
    Add OPT$(\tau_{i+1} - 1)$ to SOL.
    Increment i.

Theorem 16. *The* DOUBLE *algorithm is* 4*-competitive for the* ONLINESEARCH *problem.*

Proof. Recall that T denotes the length of the input sequence. Let $\tau_i \leqslant T < \tau_{i+1}$. Then, the cost of the optimal solution, OPT$(T) \geqslant$ OPT(τ_i) by monotonicity, while the cost of the online solution SOL is given by:

$$\begin{aligned} &\text{OPT}(\tau_1 - 1) + \text{OPT}(\tau_2 - 1) + \ldots + \text{OPT}(\tau_{i+1} - 1) \\ &\qquad \leqslant 2 \cdot \text{OPT}(\tau_{i+1} - 1) \leqslant 4 \cdot \text{OPT}(\tau_i). \end{aligned}$$

□

3.5 ONLINESEARCH with Predictions

In the previous section, we described a simple online algorithm for the ONLINESEARCH problem. Now, we build on this algorithm to take advantage of ML predictions. For now, we do not concern ourselves with how these predictions are generated; we will address this question in the next section.

Suppose we have a prediction $\hat{T}$ for the input length T of an ONLINESEARCH problem instance. Naïvely, we might trust this prediction completely and buy the solution OPT$(\hat{T})$. While this algorithm is perfect if the prediction is accurate, it can fail in two ways if the

prediction is inaccurate: (a) if $T \ll \hat{T}$ and therefore $\text{OPT}(T) \ll \text{OPT}(\hat{T})$, then the algorithm has a large competitive ratio, and (b) if $T > \hat{T}$, then $\text{OPT}(\hat{T})$ may not even be feasible for T. A natural idea is to then progressively add $\text{OPT}(t)$ solutions for small values of t (similar to DOUBLE) until a certain threshold is reached, before buying the predicted optimal solution $\text{OPT}(\hat{T})$. Next, if $T > \hat{T}$, the algorithm can resume buying solution $\text{OPT}(t)$ for $t > \hat{T}$, again using DOUBLE, until the actual input T is reached.

One problem with this strategy, however, is that the algorithm does not degrade gracefully around the prediction, a property that we will need later. In particular, if T is only slighter larger than $\hat{T}$, then the algorithm adds a solution that has cost $2 \cdot \text{OPT}(\hat{T})$, thereby realizing the worst case scenario in Theorem 16 that was achieved without any prediction. Our work-around for this issue is to buy $\text{OPT}(t)$ for a t slightly larger than $\hat{T}$, instead of $\text{OPT}(\hat{T})$ itself, which secures us against the possibility of the actual input being slightly longer than the prediction. We call this algorithm PREDICT-AND-DOUBLE (Algorithm 7). Here, we use a hyper-parameter ε that offers a tradeoff between the consistency and robustness of the algorithm. We also use the following definition:

Definition 13. *Given an input length τ and any $\alpha > 0$, we use* $\text{MAX-LENGTH}(\alpha, \tau)$ *to denote the largest length t such that* $\text{OPT}(t) \leqslant \alpha \cdot \text{OPT}(\tau)$.

As described in the introduction, the desiderata for an online algorithm with predictions are its consistency and robustness; we establish the tradeoff between these parameters for the PREDICT-AND-DOUBLE algorithm below.

Theorem 17. *The* PREDICT-AND-DOUBLE *algorithm has a consistency of $1+\varepsilon$ and robustness of $5\left(1+\frac{1}{\varepsilon}\right)$.*

Proof. When the prediction is correct, i.e., $T = \hat{T}$, the algorithm only runs Phases 1 and 2. At the end of Phase 1, by Theorem 16, the cost of SOL is at most $4 \cdot \text{OPT}(t_1) \leqslant (4\varepsilon/5) \cdot \text{OPT}(\hat{T})$. In Phase 2, the algorithm buys a single solution of cost at most $(1+\varepsilon/5) \cdot \text{OPT}(\hat{T})$.

Algorithm 7 PREDICT-AND-DOUBLE

Input: The input sequence $\mathscr{I}$ and prediction $\hat{T}$.
Output: The online solution SOL.

Set SOL $:= \varnothing$, $t_1 :=$ MIN-LENGTH$(\varepsilon/5, \hat{T})$, and $t_2 :=$ MAX-LENGTH$(1+\varepsilon/5, \hat{T})$.
Phase 1: Execute DOUBLE while $t < t_1$.
Phase 2: At $t = t_1$, add OPT(t_2) to SOL.
Phase 3: If $t > t_2$, resume DOUBLE as follows:
Set $i := 0$, $\tau_0 := t_2 + 1$.
for $t = t_2+1, t_2+2, \ldots, T$
  if $t = \tau_i$
    Set $\tau_{i+1} =$ MIN-LENGTH$(2, \tau_i)$.
    Add OPT$(\tau_{i+1} - 1)$ to SOL.
    Increment i.

Adding the two, and noting that the optimal cost is OPT$(\hat{T})$, we get a consistency bound of $1+\varepsilon$.

For robustness, we consider three cases. First, if $t < t_1$, then the competitive ratio is 4 by Theorem 16. Next, if $t > t_2$, then the total cost of SOL is at most $(1+\varepsilon)$OPT(T) in Phases 1 and 2 (from the consistency analysis above), and at most $4 \cdot$ OPT(T) in Phase 3 by Theorem 16. Thus, in this case, the competitive ratio is $5+\varepsilon$. Finally, we consider the case $t_1 \leqslant t \leqslant t_2$. Here, the algorithm runs Phases 1 and 2, and the cost of SOL is at most $(1+\varepsilon) \cdot$ OPT$(\hat{T})$ by the consistency analysis above. By monotonicity, the optimal solution is smallest when $T = t_1$, i.e., OPT$(T) \geqslant \frac{\varepsilon}{5} \cdot$ OPT$(\hat{T})$. Thus, the competitive ratio is bounded by $5\left(1+\frac{1}{\varepsilon}\right)$. □

We also show that this tradeoff between $(1+\varepsilon)$-consistency and $O(1/\varepsilon)$-robustness bounds is essentially tight.

Theorem 18. *Any algorithm for the* ONLINESEARCH *problem with predictions that has a consistency bound of* $1+\varepsilon$ *must have a robustness bound of* $\Omega\left(\frac{1}{\varepsilon}\right)$.

Proof. If $T \geqslant \hat{T}$, the algorithm has to buy a solution that is feasible for $\hat{T}$ at some time $\tau \leqslant \hat{T}$. In particular, we must have OPT$(\tau) \leqslant \varepsilon \cdot$ OPT$(\hat{T})$ for deterministic algorithms,

else the consistency bound would be $> 1 + \varepsilon$ simply based on being feasible for $t = \tau$ which incurs cost $> \varepsilon \cdot \text{OPT}(\hat{T})$ and again for $t = \hat{T}$ which incurs an additional cost of $\text{OPT}(\hat{T})$. This implies a robustness bound of $\Omega\left(\frac{1}{\varepsilon}\right)$ if the input $T = \tau$. The same argument extends to randomized algorithms: now, since $\mathbb{E}[\text{OPT}(\tau)] \leqslant \varepsilon \cdot \text{OPT}(\hat{T})$, it follows that $\mathbb{E}\left[\frac{\text{OPT}(\hat{T})}{\text{OPT}(\tau)}\right] \geqslant \frac{\text{OPT}(\hat{T})}{\mathbb{E}[\text{OPT}(\tau)]} = \Omega\left(\frac{1}{\varepsilon}\right)$.

□

Having shown the consistency and robustness of the PREDICT-AND-DOUBLE algorithm, we now analyze how its competitive ratio varies with error in the prediction $\hat{T}$. In particular, the next lemma shows that the competitive ratio gracefully degrades with prediction error for small error, and is capped at 4 for large error.

Lemma 9. *Given a prediction $\hat{T}$ for the input length, the competitive ratio of* PREDICT-AND-DOUBLE *is given by:*

$$\text{CR} \leqslant \begin{cases} 4\,,\ T \leqslant t_1 \\ (1+\varepsilon) \cdot \frac{\text{OPT}(\hat{T})}{\text{OPT}(T)}\,,\ t_1 \leqslant T \leqslant t_2 \\ 4\,,\ T > t_2 \end{cases}$$

where t_1 represents the minimum value of t that satisfies $\text{OPT}(t) \geqslant \frac{\varepsilon}{5} \cdot \text{OPT}(\hat{T})$ and t_2 represents the maximum value of t that satisfies $\text{OPT}(t) \leqslant (1 + \frac{\varepsilon}{5}) \cdot \text{OPT}(\hat{T})$.

Proof. When $T \leqslant t_1$, the competitive ratio of 4 follows from the doubling strategy of the algorithm. Next, when $t_1 < T \leqslant t_2$, the algorithm pays at most $4 \cdot \frac{\varepsilon}{5} \cdot \text{OPT}(\hat{T})$ until $t = t_1$ and then pays at most $\left(1 + \frac{\varepsilon}{5}\right) \cdot \text{OPT}(\hat{T})$ for the solution $\text{OPT}(t_2)$, which adds up to at most $(1+\varepsilon) \cdot \text{OPT}(\hat{T})$. In contrast, the optimal cost is $\text{OPT}(T)$; hence, the competitive ratio is $(1+\varepsilon) \cdot \frac{\text{OPT}(\hat{T})}{\text{OPT}(T)}$. Finally, when $T > t_2$, then let $\tau_j \leqslant T < \tau_{j+1}$ (using the notation in Algorithm 7). The algorithm pays at most

$$\left(1 + \varepsilon + 2 + \ldots 2^{j+1}\right) \text{OPT}(\hat{T}) \leqslant 2^{j+2} \cdot \text{OPT}(\hat{T}),$$

while the optimal cost is at least $2^j \cdot \text{OPT}(\hat{T})$. Hence, the competitive ratio is at most 4. □

3.6 A Regression Approach for ONLINESEARCH

In the previous section, we designed an algorithm for the ONLINESEARCH problem that utilizes ML predictions. Now we delve deeper into how we can generate these predictions. More generally, we develop a regression-based approach to learn to solve an ONLINESEARCH problem.

Recall that an online algorithm constitutes a sequence of solutions that the algorithm buys at different times of the input sequence (see Algorithm 8 for a generic description of a LEARNTOSEARCH algorithm).

Algorithm 8 A general LEARNTOSEARCH algorithm

Training: Given a Sample Set S, the training phase outputs a mapping M from every feature vector $x \in \mathbb{X}$ to an increasing sequence of positive integers
Testing: Given unknown sample $x \in \mathbb{X}$, define thresholds $M(x) = (\tau_0, \tau_1 \ldots)$
Set $i := 0, \text{SOL} := \text{OPT}(\tau_0 - 1)$.
while (Input has not ended)
  if (SOL is infeasible)
    $\text{SOL} := \text{OPT}(\tau_{i+1} - 1)$.
    Increment i.

3.6.1 The Sample Complexity of LEARNTOSEARCH

Our overall strategy is to learn a suitable predictor function $f \in \mathscr{F}$ and use $f(x)$ as a prediction in the PREDICT-AND-DOUBLE algorithm. Note that prediction errors on the two sides (over- and under-estimation) affect the competitive ratio of PREDICT-AND-DOUBLE (given by Lemma 9) in different ways. If we underestimate $\text{OPT}(T)$ by a factor less than $1 + \frac{\varepsilon}{5}$, i.e., $\text{OPT}(T') \leqslant \text{OPT}(T) \leqslant \left(1 + \frac{\varepsilon}{5}\right) \cdot \text{OPT}(T')$, the competitive ratio remains $1 + O(\varepsilon)$, but a larger underestimate causes the competitive ratio to climb up to 4. On the other hand, if we overestimate $\text{OPT}(T)$, then the competitive ratio grows steadily by the ratio of over-estimation, until it reaches $5 \cdot \left(1 + \frac{1}{\varepsilon}\right)$, beyond which it drops down (and stays at) 4. This asymmetric dependence is illustrated in Figure 3.1.

At a high level, our goal is to use regression to obtain the best function $f \in \mathscr{F}$. But, the

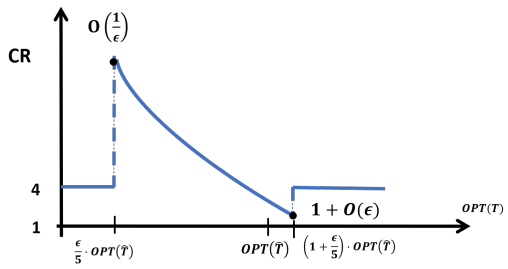

FIGURE 3.1: Competitive ratio of the PREDICT-AND-DOUBLE algorithm for a fixed prediction $\hat{T}$ as a function of the input T, where the prediction is $\hat{T}$

asymmetric behavior of the competitive ratio suggests that we should not use a standard loss function in the regression analysis. Let ε be the accuracy parameter for the PREDICT-AND-DOUBLE algorithm, and let $\hat{y} = \ln \text{OPT}(\hat{T})$ and $y = \ln \text{OPT}(T)$ be the predicted and actual log-cost of the optimal solution respectively. Then we define the following loss function that follows the asymmetric behaviour of the competitive ratio for PREDICT-AND-DOUBLE:

Definition 14. *The ε-parameterized competitive error is defined as:*

$$\ell_\varepsilon(y,\hat{y}) = \begin{cases} \frac{5}{\varepsilon} - 1 \textit{ when } y \leqslant \hat{y} - \ln\frac{5}{\varepsilon} \\ e^{\hat{y}-y} - 1 \textit{ when } \hat{y} - \ln\frac{5}{\varepsilon} < y \leqslant \hat{y} \\ \frac{1}{\varepsilon}\cdot(y-\hat{y}) \textit{ when } \hat{y} < y \leqslant \hat{y} + \ln\left(1+\frac{\varepsilon}{5}\right) \\ 1 \textit{ when } y > \hat{y} + \ln\left(1+\frac{\varepsilon}{5}\right). \end{cases}$$

We give more justification for using this loss function, and show that standard loss functions do not suffice for our purposes in Section 3.7. Using this loss function, we can measure the error of a function for an input distribution or for a fixed input set:

Definition 15. *Given a distribution $\mathbb{D}$ on the set $\mathbb{X}\times\mathbb{Y}$ and function $f:\mathbb{X}\mapsto\mathbb{Y}$, we define*

$$\mathbf{ER}_{\mathbb{D},\varepsilon}(f) = \mathbb{E}_{(x,y)\sim\mathbb{D}}[\ell_\varepsilon(y,f(x))].$$

Alternatively, for a set of samples, $S\sim\mathbb{D}^m$, we define,

$$\mathbf{ER}_{S,\varepsilon}(f) = \frac{1}{m}\cdot\sum_{i=1}^{m}\ell_\varepsilon(y_i,f(x_i)).$$

Our high-level goal is to use samples to optimize for the loss function called ε-parameterized competitive error that we defined above over the function class $\mathscr{F}$, and then use an algorithm that translates the empirical error bound to a competitive ratio bound. This requires, in the training phase, that we optimize the empirical loss on the training samples. We define such a minimizer below:

Definition 16. *For a given set of samples $S \sim \mathbb{D}$ and a function family $\mathscr{F}$, we denote an optimization scheme $\mathscr{O} : S \mapsto \mathscr{F}$ as $\varepsilon-$Sample Error Minimizing (SEM) if it returns a function $\hat{f} \in \mathscr{F}$ satisfying:*

$$\mathbf{ER}_{S,\varepsilon}(\hat{f}) \leqslant \inf_{f \in \mathscr{F}} \left[\mathbf{ER}_{S,\varepsilon}(f)\right] + \varepsilon.$$

For the rest of this chapter, we will assume that we are given an $\varepsilon-$ SEM routine for arbitrary $\varepsilon > 0$. We are now ready to present our LEARNTOSEARCH algorithm (Algorithm 9), which basically uses the predictor with minimum expected loss to make predictions for PREDICT-AND-DOUBLE.

Algorithm 9 A LEARNTOSEARCH algorithm with accuracy parameter ε

Training:
Input: Sample Set S, Function Family $\mathscr{F}$
Output: $\hat{f}$ output by an ε-SEM algorithm $\mathscr{O}$, i.e., $\mathbf{ER}_{S,\varepsilon}(\hat{f}) \leqslant \inf_{\tilde{f} \in \mathscr{F}} \mathbf{ER}_{S,\varepsilon}(\tilde{f}) + \varepsilon$.
Testing:
Given new sample x, set $\hat{y} = \hat{f}(x)$.
Predicted prefix length: $\hat{T} = \max_{\mathrm{OPT}(t) \leqslant e^{\hat{y}}} t$.
Call PREDICT-AND-DOUBLE with $\hat{T}$ and ε.

We relate the competitive ratio of Algorithm 9 to the error of function $\hat{f}$ obtained during training:

Lemma 10. *Algorithm 9 has a competitive ratio upper bounded by $\left(1 + \varepsilon + 3 \cdot \mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f})\right)$.*

Proof. We use Lemma 9 to prove this result. Let $\hat{y}$ and $\hat{T}$ be as in the description of Algorithm 9. Let t_1, t_2 be as in the statement of Lemma 9 with respect to $\hat{T}$.

Now consider the following cases (as in the statement of Lemma 9)

- $T < t_1$: by definition of t_1, $\text{OPT}(T) < \frac{\varepsilon}{5} \cdot \text{OPT}(\hat{T})$, and so, $y < \hat{y} - \ln \frac{5}{\varepsilon}$. Since the competitive ratio is at most 4 in this case, we see that (using Definition 14) this is at most $\ell_\varepsilon(\hat{y}, y)$.

- $t_1 \leqslant T \leqslant \hat{T}$: In this case, $\hat{y} - \ln \frac{5}{\varepsilon} \leqslant y \leqslant \hat{y}$. This case, the competitive ratio is at most

$$(1+\varepsilon) \cdot e^{\hat{y}-y} \leqslant 1 + \varepsilon + 3\left(e^{\hat{y}-y} - 1\right) = 1 + \varepsilon + 3\ell_\varepsilon(\hat{y}, y),$$

  where the first inequality follows from the fact that $\hat{y} \geqslant y$.

- $\hat{T} \leqslant T \leqslant t_2$: Here $\hat{y} < y \leqslant \hat{y} + \ln\left(1 + \frac{\varepsilon}{5}\right)$. Again, the competitive ratio is at most

$$(1+\varepsilon) \cdot e^{\hat{y}-y} \leqslant 1 + \varepsilon \leqslant 1 + \varepsilon + 3\ell_\varepsilon(\hat{y}, y),$$

  where the first inequality follows from $\hat{y} < y$.

- $T > t_2$: Here $y \geqslant \hat{y} + \ln\left(1 + \frac{\varepsilon}{5}\right)$. Lemma 9 shows that the competitive ratio is at most 4, which is at most $1 + \varepsilon + 3\ell_\varepsilon(\hat{y}, y)$.

We note that for all values y, the competitive ratio is upper bounded by $1 + \varepsilon + 3 \cdot \ell_\varepsilon(\hat{y}, y)$, where $\ell_\varepsilon(\hat{y}, y)$ is the ε-parameterized competitive error of $\hat{y}$. So, the expected competitive ratio is $\leqslant 1 + \varepsilon + 3 \cdot \mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f})$. □

Standard and Agnostic Models. We consider two different settings. First, we assume that the function class $\mathscr{F}$ contains the function f^* that maps the feature set x to y – we call this the *standard* model. We relax this assumption in the more general *agnostic* model, where the function class $\mathscr{F}$ is arbitrary. In terms of the error function, in the standard model, we have $\inf_{f \in \mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f) = \inf_{f \in \mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f) = 0$, while no such guarantee holds in the agnostic model.

3.6.2 Analysis in the Standard Model

Next, we analyze the competitive ratio of Algorithm 9 in the standard model, i.e., when $\inf_{f \in \mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f) = \inf_{f \in \mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f) = 0$.

Theorem 19. *In the standard model, Algorithm 9 obtains a competitive ratio of* $1+O(\varepsilon)$ *with probability at least* $1-\delta$*, when using* $O\left(\frac{H\cdot d\log\frac{1}{\varepsilon}\log\frac{1}{\delta}}{\varepsilon}\right)$ *samples, where* $d = Pdim(\mathscr{F})$.

When the cost of the optimal solution $\textsc{opt}(\tau)$ is hard to compute, we can replace the offline optimal with an online algorithm that achieves competitive ratio c given the value of τ to get the following:

Corollary 4. *In the standard model, if there exists a c-competitive algorithm for* $\textsc{opt}(\tau)$ *given the value of prefix-length* τ*, Algorithm 9 obtains a competitive ratio of* $c(1+O(\varepsilon))$ *with probability at least* $1-\delta$*, when using* $O\left(\frac{H\cdot d\log\frac{1}{\varepsilon}\log\frac{1}{\delta}}{\varepsilon}\right)$ *samples, where* $d = Pdim(\mathscr{F})$.

We also show that the result in Theorem 19 is tight up to a factor of $H\log 1/\varepsilon$:

Theorem 20. *Let* $\mathscr{F}$ *be a family of real valued functions such that there exists a function* $f^* : \mathbb{X} \mapsto \mathbb{Y}$ *that* $f^*(x) = y$ *and let* $d = Pdim(\mathscr{F})$*. There exists an instance of the* LEARN-TOSEARCH *problem that enforces any algorithm to query* $\Omega\left(\frac{d\log\frac{1}{\delta}}{\varepsilon}\right)$ *samples in order to have an expected competitive ratio of* $1+\varepsilon$ *with probability* $\geqslant 1-\delta$.

In order to have sample complexity bounds relating to the pseudo dimension of the function class, we would need to introduce the notion of covering numbers and relate them to the pseudo-dimension.

Definition 17. *Given a set S in Euclidean space and a metric* $d(\cdot,\cdot)$*, the set* $W \subseteq S$ *is said to be* ε *cover of S if for any* $s \in S$*, there exists a* $w \in W$ *such that* $d(s,w) \leqslant \varepsilon$*. The smallest possible cardinality of such an* ε *cover is known as the* ε *covering number of S with respect to d and is denoted as* $\mathscr{N}_{d(\cdot,\cdot)}(\varepsilon, S)$.

When d is given by the distance metric

$$d_p(r,s) = \Big|\sum_{i=1}^{d}(r_i - s_i)^p\Big|^{1/p},$$

where $r = (r_1, r_2 \dots r_d), s = (s_1, s_2 \dots s_d) \in \mathbb{R}^d$, we shall denote the ε covering number of a set S by $\mathscr{N}_p(\varepsilon, S)$. For a given real-valued function family $\mathscr{F}$ and $x = (x_1, x_2, \dots, x_m) \in \mathbb{X}^m$, we denote

$$\mathscr{F}_{|x} = \{(f(x_1), f(x_2), \dots, f(x_m)) \mid f \in \mathscr{F}\} \quad \text{and}$$

$$\mathscr{N}_p(\varepsilon, \mathscr{F}, m) = \sup_{x \in \mathbb{X}^m} \left[\mathscr{N}_p(\varepsilon, \mathscr{F}_{|x})\right].$$

Note that $\mathscr{N}_1(\varepsilon, \mathscr{F}, m) \leqslant \mathscr{N}_2(\varepsilon, \mathscr{F}, m) \leqslant \mathscr{N}_\infty(\varepsilon, \mathscr{F}, m)$.

Given a loss function $\ell(\cdot, \cdot)$, and sample set $S = \{(x_i, y_i), i = 1, 2 \dots m\}$, we can define $(\ell_{\mathscr{F}})_{|S}$ as :

$$(\ell_{\mathscr{F}})_{|S} = \{\ell_f(x_i, y_i) | (x_i, y_i) \in S, f \in \mathscr{F}\} \subset \mathbb{R}^m.$$

where $\ell_f(x_i, y_i) = \ell(y_i, f(x_i))$.

The following is a well-known result that relates covering numbers to the pseudo dimension (cf. Theorem 12.2 in Book Anthony and Bartlett (1999)):

Lemma 11. *Let $\mathscr{F}$ be a real-valued function family with pseudo dimension d, then for any $\varepsilon \leqslant \frac{1}{d}$, we have*

$$\mathscr{N}_1(\varepsilon, \mathscr{F}, m) \leqslant O\left(\frac{1}{\varepsilon^d}\right).$$

First, we relate covering numbers to the difference between $\mathbf{ER}_{S,\varepsilon}(f)$ and $\mathbf{ER}_{\mathbb{D},\varepsilon}(f)$. This will be crucial in proving Theorem 19.

Lemma 12. *Let $\mathbb{D}$ be a distribution on $\mathbb{X} \times \mathbb{Y}$ and let $S \in \mathbb{D}^m$. For $0 \leqslant \eta \leqslant 12$ and $m \geqslant \frac{8 \cdot H}{\eta^2}$, for any real valued function family $\mathscr{F}$, we have:*

$$\mathbb{P}_{S \in \mathbb{D}^m}\left[\mathbf{ER}_{\mathbb{D},\varepsilon}(f) \leqslant (1+\alpha) \cdot \mathbf{ER}_{S,\varepsilon}(f) + \eta^2 \cdot \left(1 + \frac{1}{\alpha}\right)\right] \leqslant 4 \cdot \mathscr{N}_1\left(\frac{\eta\varepsilon}{8}, \mathscr{F}, 2m\right) \cdot \exp\left(-\frac{m \cdot \eta^2 \cdot}{64H}\right).$$

Moreover, we also have the other side as :

$$\mathbb{P}_{S \in \mathbb{D}^m}\left[\mathbf{ER}_{S,\varepsilon}(f) \leqslant \frac{2\alpha+1}{(1+\alpha)} \cdot \mathbf{ER}_{\mathbb{D},\varepsilon}(f) + \eta^2 \cdot \left(1 + \frac{1}{\alpha}\right)\right] \leqslant 4 \cdot \mathscr{N}_1\left(\frac{\eta\varepsilon}{8}, \mathscr{F}, 2m\right) \cdot \exp\left(-\frac{m \cdot \eta^2 \cdot}{64H}\right)$$

To prove this lemma, we need the following definition.

Definition 18. *The normalised (ε parameterised) error is defined as :*

$$\hat{\mathbf{ER}}_{S,\mathbb{D},\varepsilon}(f) = \frac{|\mathbf{ER}_{S,\varepsilon}(f) - \mathbf{ER}_{\mathbb{D},\varepsilon}(f)|}{\sqrt{\mathbf{ER}_{\mathbb{D},\varepsilon}(f)}}.$$

Lemma 13. *Let $\mathbb{D}$ be a distribution on $\mathbb{X} \times \mathbb{Y}$ and let $S \sim \mathbb{D}^m$. For $\eta \leqslant 12$, and $m \geqslant \frac{8H}{\eta^2}$, for any real valued function family $\mathscr{F}$, we have*

$$\mathbb{P}_{S\sim\mathbb{D}^m}\left[\sup_{f\in\mathscr{F}} |\hat{\mathbf{ER}}_S(f) - \hat{\mathbf{ER}}_{\mathbb{D}}(f)| \geqslant \eta\right] \leqslant 4 \cdot \mathscr{N}(\eta/8, \ell_{\mathscr{F}}, 2m) \cdot exp\left(-\frac{m\cdot\eta^2}{64H}\right).$$

We break this proof into four separate claims as illustrated below.

First, we reduce the probability of the event: $[\hat{\mathbf{ER}}_S(f) \geqslant \eta]$ to a probability term involving two sample sets $S, \bar{S}$ the members of which are drawn independently.

Lemma 14. *For $m \geqslant \frac{8H}{\eta^2}$, we have that:*

$$\mathbb{P}_{S\sim\mathbb{D}^m}\left[\sup_{f\in\mathscr{F}} \hat{\mathbf{ER}}_S(f) \geqslant \eta\right] \leqslant 2 \cdot \mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\left[\sup_{f\in\mathscr{F}} |\hat{\mathbf{ER}}_S(f) - \hat{\mathbf{ER}}_{\bar{S}}(f)| \geqslant \eta/2\right].$$

Proof. For a given sample $S \sim \mathbb{D}^m$, let $f^S_{bad} \in \mathscr{F}$ denote a function f such that $|\hat{\mathbf{ER}}_S(f) - \hat{\mathbf{ER}}_{\mathbb{D}}(f)| \geqslant \eta$ if it exists, otherwise we set f to be any fixed function in the family $\mathscr{F}$. Now,

$$\begin{aligned}
\mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\left[\sup_{f\in\mathscr{F}} |\hat{\mathbf{ER}}_S(f) - \hat{\mathbf{ER}}_{\bar{S}}(f)| \geqslant \frac{\eta}{2}\right] &\geqslant \mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\left[|\hat{\mathbf{ER}}_S(f^S_{bad}) - \hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad})| \geqslant \frac{\eta}{2}\right] \\
&\geqslant \mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\left[\left\{\hat{\mathbf{ER}}_S(f^S_{bad}) \geqslant \eta\right\} \cap \left\{\hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad}) \leqslant \frac{\eta}{2}\right\}\right] \\
&= \mathbb{P}_{S\sim\mathbb{D}^m}\left[\hat{\mathbf{ER}}_S(f^S_{bad}) \geqslant \eta\right] \cdot \mathbb{P}_{\bar{S}\sim\mathbb{D}^m|S}\left[\hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad}) \leqslant \frac{\eta}{2}\right] \\
&= \mathbb{P}_{S\sim\mathbb{D}^m}\left[\sup_{f\in\mathscr{F}} \hat{\mathbf{ER}}_S(f) \geqslant \eta\right] \cdot \mathbb{P}_{\bar{S}\sim\mathbb{D}^m|S}\left[\hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad}) \leqslant \frac{\eta}{2}\right].
\end{aligned}$$

Now, the term $\mathbb{P}_{\bar{S}\sim\mathbb{D}^m|S}\left[\hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad}) \leqslant \frac{\eta}{2}\right]$ is bounded below by the Chebyshev's inequality as follows:

$$\mathbb{P}_{\bar{S}\sim\mathbb{D}^m|S}\left[\hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad}) \leqslant \frac{\eta}{2}\right] \geqslant 1 - \frac{\mathrm{Var}_{\bar{S}\sim\mathbb{D}^m|S}\left[\hat{\mathbf{ER}}_{\bar{S}}(f^S_{bad})\right]}{\frac{\eta^2}{4}} \geqslant 1 - \left(\frac{H}{m\cdot\frac{\eta^2}{4}}\right) \geqslant \frac{1}{2},$$

where the last inequality follows from the fact that $m \geqslant \left(\frac{8\cdot H}{\eta^2}\right)$ and that

$$\frac{Var(\mathbf{ER}_{\bar{S}}(f^S_{bad}))}{\mathbf{ER}_{\mathbb{D}}(f^S_{bad})} = \frac{1}{m}\cdot\sum_{i=1}^{i=m}\frac{Var(\ell_{f^S_{bad}}(x_i,y_i)))}{\mathbb{E}(\ell_{f^S_{bad}}(x_i,y_i))} \leqslant \frac{1}{m}\cdot H,$$

because $\ell_{f^S_{bad}}(x_i,y_i)) \in [0,H]$.

□

The second claim intuitively says that the probabilities remain unchanged under symmetric permutations. Let σ denote a permutation on the set $\{1,2,\ldots,2m\}$ such that for each $i \in \{1,2,\ldots,m\}$, we use either of the two mappings:

- $\sigma(i) = i$ and $\sigma(m+i) = m+i$, or
- $\sigma(i) = m+i$ and $\sigma(m+i) = i$.

Let Γ^m denote the set of all such permutations σ. Suppose we draw i.i.d. samples $S \sim \mathbb{D}^m$ and $\bar{S} \sim \mathbb{D}^m$; let $S = \{s_1, s_2, \ldots, s_m\}$ and $\bar{S} = \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_m\}$. Then, define $\sigma(S)$ and $\sigma(\bar{S})$ by using a permutation $\sigma \in \Gamma^m$ as follows. Let $\sigma(S) = \{s'_1, s'_2, \ldots, s'_m\}$ and $\sigma(\bar{S}) = \{\bar{s}'_1, \bar{s}'_2, \ldots, \bar{s}'_m\}$ such that $s'_i = s_i$ and $\bar{s}'_i = \bar{s}_i$ if $\sigma(i) = i$ and $\sigma(m+i) = m+i$, while $s'_i = \bar{s}_i$ and $\bar{s}'_i = s_i$ if $\sigma(i) = m+i$ and $\sigma(m+i) = i$. Let U^m denote the uniform distribution over Γ^m.

Lemma 15. *For every $f \in \mathscr{F}$:*

$$\mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\left[|\hat{\mathbf{ER}}_S(f) - \hat{\mathbf{ER}}_{\bar{S}}(f)| \geqslant \eta/2\right] \leqslant \sup_{(S,\bar{S})\in(\mathbb{X}\times\mathbb{Y})^{2m}}\left(\mathbb{P}_{\sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \geqslant \eta/2\right]\right).$$

Proof. We have for every $f \in \mathscr{F}$:

$$\begin{aligned}
&\mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\left[|\hat{\mathbf{ER}}_S(f) - \hat{\mathbf{ER}}_{\bar{S}}(f)| \geqslant \eta/2\right] \\
&\qquad = \mathbb{P}_{(S,\bar{S})\sim D^m\times D^m,\ \sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \geqslant \eta/2\right] \quad \text{(by the i.i.d. property)} \\
&\qquad \leqslant \sup_{(S,\bar{S})\in(\mathbb{X}\times\mathbb{Y})^{2m}}\left(\mathbb{P}_{\sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \geqslant \eta/2\right]\right),
\end{aligned}$$

where in the last expression we chose the members of $S, \bar{S}$ adversarially instead of randomly.

□

Third, we make use of covering numbers to quantify the above probability.

Lemma 16. *Fix a $(S,\bar{S}) \in (\mathbb{X}\times\mathbb{Y})^{2m}$. Consider the set $\mathscr{G} \in \mathscr{F}$ such that $\ell_{\mathscr{G}}(S,\bar{S})$ is an $\frac{\eta}{8}$-covering (wrt $d_1(\cdot,\cdot)$) of the set $\ell_{\mathscr{F}}(S,\bar{S}) = \{\ell_f(x_i,y_i) \mid (x_i,y_i) \in S\cup\bar{S}, f\in\mathscr{F}\} \subset [0,H]^{2m}$. Then :*

$$\mathbb{P}_{S\sim\mathbb{D}^m}\left[\sup_{f\in\mathscr{F}} \hat{\mathbf{ER}}_S(f) \geqslant \eta\right] \leqslant \mathscr{N}\left(\frac{\eta}{8}, \ell_{\mathscr{F}}, 2m\right)\cdot\max_{g\in\mathscr{G}}\mathbb{P}_{S\sim\mathbb{D}^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \geqslant \frac{\eta}{4}\right].$$

Proof. Note that the cardinality of $\mathscr{G}$ is less than $\mathscr{N}_1(\eta/8, \ell_{\mathscr{F}}, 2m)$ and is a bounded number. We claim that whenever an $f \in \mathscr{F}$ satisfies, $|\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \geqslant \frac{\eta}{2}$, then there exists a $g \in \mathscr{G}$ such that, $|\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \geqslant \frac{\eta}{4}$.

Let g satisfy that, $\frac{1}{2m}[\sum_{i=1}^{2m}|\ell_g(x_i,y_i) - \ell_g(x_i,y_i)|] \leqslant \frac{\eta}{8}$.

We are guaranteed that such a g exists, since it is in the cover.

$$\begin{aligned}
\frac{\eta}{2} &\leqslant |\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \\
&= |(\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(S)}(g)) - (\hat{\mathbf{ER}}_{\sigma(\bar{S})}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)) + (\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g))| \\
&= |(\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(S)}(g))| + |\hat{\mathbf{ER}}_{\sigma(\bar{S})}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| + |\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \\
&= |\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| + |\frac{1}{m}\sum_{i=1}^{m}\big(\ell_f(x_i,y_i) - \ell_g(x_i,y_i)\big)| + |\frac{1}{m}\sum_{i=m+1}^{2m}\big(\ell_f(x_i,y_i) - \ell_g(x_i,y_i)\big)| \\
&\leqslant |\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| + \frac{1}{m}\sum_{i=1}^{2m}|\ell_f(x_{\sigma(i),y_{\sigma(i)}}) - \ell_g(x_{\sigma(i),y_{\sigma(i)}})| \\
&< |\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| + \frac{\eta}{4}.
\end{aligned}$$

Therefore, we get:

$$\begin{aligned}
&\mathbb{P}_{S\sim\mathbb{D}^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \geqslant \eta/2\right] \\
&\qquad\leqslant \mathbb{P}_{S\sim\mathbb{D}^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \geqslant \eta/4\right] \\
&\qquad\leqslant |\mathscr{G}| \cdot \max_{g\in\mathscr{G}} \mathbb{P}_{S\sim\mathbb{D}^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \geqslant \eta/4\right]. \\
&\qquad\leqslant \mathscr{N}\left(\frac{\eta}{8}, \ell_{\mathscr{F}}, 2m\right) \cdot \max_{g\in\mathscr{G}} \mathbb{P}_{S\sim\mathbb{D}^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \geqslant \eta/4\right]. \qquad \square
\end{aligned}$$

Our final step is to bound $\mathbb{P}_{\sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(g) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)| \geqslant \frac{\eta}{4}\right]$ for all $(S,\bar{S}) \in (\mathbb{X}\times\mathbb{Y})^{2m}$, which is effected by the last claim.

Lemma 17. *For any $f \in \mathscr{F}$, with $\eta \leqslant 12$:*

$$\mathbb{P}_{\sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(f) - \hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)| \geqslant \frac{\eta}{4}\right] \leqslant 2\cdot\exp\left(-\frac{m\cdot\eta^2}{64H}\right).$$

Proof. We use Bernstein's inequality Craig (1933) that says for n independent zero-mean

random variables X_i's satisfying $|X_i| \leqslant M$, we have:

$$\mathbb{P}\left(|\sum_i^n X_i| > t\right) \leqslant 2 \cdot \exp\left(\frac{-\frac{t^2}{2}}{\sum_{i=1}^n \mathbb{E}[X_i^2] + \frac{1}{3}M \cdot t}\right).$$

Note that the quantity $\mathbf{E}\hat{\mathbf{R}}_{\sigma(S)}(f) - \mathbf{E}\hat{\mathbf{R}}_{\sigma(\bar{S})}(f)$ is simply an average of m random variables, each of which has a variance upper bounded by H. Then applying the above bound:

$$\begin{aligned}
&\mathbb{P}_{\sigma \sim U^m}\left[|\mathbf{E}\hat{\mathbf{R}}_{\sigma(S)}(f) - \mathbf{E}\hat{\mathbf{R}}_{\sigma(\bar{S})}(f)| \geqslant \frac{\eta}{4}\right] \\
&\leqslant 2 \cdot \exp\left(-\frac{m \cdot \eta^2}{32H(1+\frac{\eta}{12})}\right) \\
&\leqslant 2 \cdot \exp\left(-\frac{m \cdot \eta^2}{64H}\right).
\end{aligned}$$

□

We will use the Lipschitz property of the loss function to relate the covering numbers of $\ell_{\mathscr{F}}$ and $\mathscr{F}$ as follows:

Lemma 18. *Let $\ell : \mathbb{Y} \times \mathbb{Y} \mapsto [0,H]$ be a loss function such that it satisfies:*

$$|\ell(y_1,y) - \ell(y_2,y)| \leqslant L \cdot |y_1 - y_2|.$$

Then, for any real valued function family $\mathscr{F}$, we have:

$$\mathscr{N}(\varepsilon, \ell_{\mathscr{F}}, m) \leqslant \mathscr{N}\left(\frac{\varepsilon}{L}, \mathscr{F}, m\right).$$

Proof. Let $S = \{(x_1,y_1)\ldots(x_m,y_m)\} \in (\mathbb{X} \times \mathbb{Y})^m$, and let $g,h \in \mathscr{F}$ be two functions. We have:

$$\frac{1}{m}\sum_{i=1}^m |\ell_g(x_i,y_i) - \ell_h(x_i,y_i)| = \frac{1}{m}\sum_{i=1}^m |\ell(y_i,g(x_i)) - \ell(y_i,h(x_i))| \leqslant \frac{L}{m}\sum_{i=1}^m |g(x_i) - h(x_i)|.$$

Hence, any $\frac{\varepsilon}{L}$ cover for $\mathscr{F}_{|x_1^m}$ is an ε cover for $(\ell_{\mathscr{F}})_{|S}$. □

Now we are ready for the proof of Lemma 13.

Proof of Lemma 13. We have:

$$\mathbb{P}_{S\sim\mathbb{D}^m}\left[\sup_{f\in\mathscr{F}}|\hat{\mathbf{ER}}_S(f)-\hat{\mathbf{ER}}_{\mathbb{D}}(f)|\geqslant\eta\right]$$

$$\leqslant 2\cdot\mathbb{P}_{(S,\bar{S})\sim D^m\times D^m}\big[\sup_{f\in\mathscr{F}}\left(|\hat{\mathbf{ER}}_S(f)-\hat{\mathbf{ER}}_{\bar{S}}(f)|\geqslant\eta/2\right)\text{ (by Lemma 14)}$$

$$\leqslant 2\cdot\sup_{(S,\bar{S})\in(\mathbb{X}\times\mathbb{Y})^{2m}}\left(\mathbb{P}_{\sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(f)-\hat{\mathbf{ER}}_{\sigma(\bar{S})}(f)|\geqslant\eta/2\right]\right)\text{ (by Lemma 15)}$$

$$\leqslant 2\cdot\mathscr{N}\left(\eta/8,\ell_{\mathscr{F}},2m\right)\cdot\max_{g\in\mathscr{G}}\left(\mathbb{P}_{\sigma\sim U^m}\left[|\hat{\mathbf{ER}}_{\sigma(S)}(g)-\hat{\mathbf{ER}}_{\sigma(\bar{S})}(g)|\geqslant\eta/4\right]\right)\text{ (by Lemma 16)}$$

$$\leqslant 4\cdot\mathscr{N}\left(\eta/8,\ell_{\mathscr{F}},2m\right)\cdot exp\left(-\frac{m\cdot\eta^2}{64H}\right)\text{ (by Lemma 17).}$$

□

Finally, we arrive at the proof of Lemma 12.

Proof of Lemma 12. Given that, $\left|\mathbf{ER}_{\mathbb{D},\varepsilon}(f)-\mathbf{ER}_{S,\varepsilon}(f)\right|\leqslant\eta\cdot\sqrt{\mathbf{ER}_{\mathbb{D},\varepsilon}(f)}$, we claim:

$$\mathbf{ER}_{\mathbb{D},\varepsilon}(f)\leqslant(1+\alpha)\cdot\mathbf{ER}_{S,\varepsilon}(f)+\left(1+\frac{1}{\alpha}\right)\cdot\eta^2$$

, and

$$\mathbf{ER}_{S,\varepsilon}(f)\leqslant\frac{2\alpha+1}{(1+\alpha)}\cdot\mathbf{ER}_{\mathbb{D},\varepsilon}(f)+\left(1+\frac{1}{\alpha}\right)\cdot\eta^2$$

To show this, we consider the following two cases:

1. If $\mathbf{ER}_{\mathbb{D},\varepsilon}(f)\leqslant\left(1+\frac{1}{\alpha}\right)^2\cdot\eta^2$, then we have $\left|\mathbf{ER}_{\mathbb{D},\varepsilon}(f)-\mathbf{ER}_{S,\varepsilon}(f)\right|\leqslant\left(1+\frac{1}{\alpha}\right)\cdot\eta^2$.

2. Otherwise, we have $\mathbf{ER}_{\mathbb{D},\varepsilon}(f)>\left(1+\frac{1}{\alpha}\right)^2\cdot\eta^2$, and we get $\mathbf{ER}_{\mathbb{D},\varepsilon}(f)\leqslant\mathbf{ER}_{S,\varepsilon}(f)+\frac{\alpha}{1+\alpha}\cdot\mathbf{ER}_{\mathbb{D},\varepsilon}(f)$, and $\mathbf{ER}_{S,\varepsilon}(f)\leqslant\frac{2\alpha+1}{(\alpha+1)}\cdot\mathbf{ER}_{\mathbb{D},\varepsilon}(f)$

In either case, both the claims follow.

Lastly, the ε parameterised loss function $\ell_\varepsilon(\cdot,\cdot)$ is $\frac{1}{\varepsilon}$-Lipschitz in its first argument, from Lemma 18, we get that:

$$\mathscr{N}\left(\frac{\eta}{8}, \ell_{\mathscr{F}}, 2m\right) \leqslant \mathscr{N}\left(\frac{\varepsilon\cdot\eta}{8}, \mathscr{F}, 2m\right).$$

□

We combine these results to present the proof of Theorem 19.

Proof of Theorem 19. By Lemma 10, it suffices to show that there exists a learning algorithm $\mathscr{L} : S^m \mapsto \mathscr{F}$ that outputs a function $\hat{f} : \mathbb{X} \mapsto \mathbb{Y} \in \mathscr{F}$ such that $\mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f}) \leqslant 4\varepsilon$. Recall that in the training phase of Algorithm 9, we use an $\varepsilon-$SEM algorithm $\mathscr{O}$ that returns a function $\hat{f}$ satisfying:

$$\mathbf{ER}_{S,\varepsilon}(\hat{f}) \leqslant \inf_{\tilde{f}\in\mathscr{F}} \mathbf{ER}_{S,\varepsilon}(\tilde{f}) + \varepsilon = \varepsilon, \tag{3.1}$$

where the last equality is because $\inf_{\tilde{f}\in\mathscr{F}} \mathbf{ER}_{S,\varepsilon}(\tilde{f}) = 0$ in the standard model. So, we are left to bound $\mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f})$ in terms of $\mathbf{ER}_{S,\varepsilon}(\hat{f})$, in particular, that $\mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f}) \leqslant 2\cdot\mathbf{ER}_{S,\varepsilon}(\hat{f}) + \varepsilon$, which would prove the theorem.

For this purpose, we employ Lemma 12. In this lemma, let us set $\alpha = 1$, and $\eta^2 = \varepsilon$ and denote the event

$$\sup_{f\in\mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f) \leqslant 4\varepsilon$$

as the "good" event; if this does not hold, we call it the "bad" event.

This leaves us to bound the probability of the bad event, which by Lemma 12, is at most

$$4\cdot\mathscr{N}_1\left(\frac{\varepsilon^{\frac{3}{2}}}{8}, \mathscr{F}, 2m\right)\cdot\exp\left(-\frac{m\cdot\varepsilon}{64H}\right).$$

This quantity is at most δ when $m \geqslant C\cdot\left(\frac{H\cdot d\cdot\log\frac{1}{\varepsilon}\cdot\log\frac{1}{\delta}}{\varepsilon}\right)$ for a large constant C, thereby proving the theorem. □

Let us now move to the lower bound : Consider the input sequence $\Sigma = \tau_0, \tau_1, \ldots$ such that $\text{OPT}(0) = 2, \text{OPT}(1) = 4$. Note that the log-cost at the two time-steps are 1 and 2 respectively. Let $\mathbb{X}$ be set of d distinct points (on the real line). Let $\mathscr{F}$ be the set of all 2^d functions from $\mathbb{X}$ to $\{0,1\}$. Clearly, the VC-dimension of $\mathscr{F}$ is given by d. For every $f \in \mathscr{F}$, we define a distribution $\mathbb{D}_f$ over pairs $(x,y) \in \mathbb{X} \times \{1,2\}$ as follows: $\mathbb{D}_f$ is the uniform distribution over $A_f := \{(x, f(x)+1) : x \in \mathbb{X}\}$. Note that this is an instance of the standard setting, because for any distribution $\mathbb{D}_f$, the corresponding function f maps x to y.

Let $\mathscr{A}$ be an algorithm for the LTS problem as above which has expected competitive ratio at most $1 + \varepsilon/4$ with probability at least $1 - \delta$. Let k be an upper bound on the sample complexity of $\mathscr{A}$. The algorithm $\mathscr{A}$, after seeing k samples, outputs a strategy. The strategy gives for each $x \in \mathbb{X}$, a probability distribution over strategies (i) and (ii) as in the previous case.

Now consider the following prediction problem $\mathscr{P}$: we choose a function f uniformly at random from $\mathscr{F}$, and are given k i.i.d. samples from $\mathbb{D}_f$. We would like to predict a function $f' \in \mathscr{F}$ which agrees with f on at least $1 - \varepsilon$ fraction of the points in $\mathbb{X}$.

Lemma 19. *Suppose the algorithm $\mathscr{A}$ has the above-mentioned properties. Then given k i.i.d. samples from an instance of $\mathscr{P}$, we can output the desired function f' with probability at least $1 - \delta$.*

Proof. Suppose the function f gets chosen. We feed the k i.i.d. samples from $\mathbb{D}_f$ to $\mathscr{A}$. The algorithm $\mathscr{A}$ outputs a strategy S which, for each x, gives a distribution $(q_x, 1 - q_x)$ over strategies (i) and (ii).

Given this strategy S, we output the desired function f' as follows. For every $x \in \mathbb{X}$, if $q_1(x) \geqslant 1/2$, we set $f'(x) = 1$, else we set it to 0. We claim that if $\mathscr{A}$ has expected competitive ratio at most $1 + \varepsilon$, then f' agrees with f on at least ε fraction of points in $\mathbb{X}$.

Suppose not. Suppose $f(x) \neq f'(x)$ for some $x \in \mathbb{X}$. If $f(x) = 0$, then the cost of the optimal strategy here is 2, whereas the algorithm $\mathscr{A}$ follows strategy (ii) with probability at

least $1/2$, and its expected cost is more than $2 \cdot \frac{1}{2} + 4 \cdot \frac{1}{2} = 3$. Similarly, if $f(x) = 1$, optimal strategy pays 4. But algorithm $\mathscr{A}$ places at least $1/2$ probability on strategy (i). Therefore, its expected cost is more than $\frac{1}{2} \cdot 4 + \frac{1}{2} \cdot 6 = 5$. In either case, it pays at least 1.25 times the optimal cost. Since f and f' disagree on at least ε-fraction of the points, it follows that the expected competitive ratio of $\mathscr{A}$ (when x is chosen uniformly from $\mathbb{X}$) is more than $1 + \varepsilon/4$, a contradiction.

Since $\mathscr{A}$ has competitive ratio at most $1 + \varepsilon/4$ with probability at least $1 - \delta$, the desired result follows. □

Now, it is well known that if we want to find a function $f' \in \mathscr{F}$ which matches with f on more than $1 - \varepsilon$ fraction of points in $\mathbb{X}$ with probability at least $1 - \delta$, we need to sample at least $\Omega\left(\frac{d}{\varepsilon} \ln\left(\frac{1}{\delta}\right)\right)$ points from $\mathbb{D}_f$ (see Thm 5.3 in Anthony and Bartlett (1999)). This proves Theorem 20.

3.6.3 Extension to the Agnostic Model

In the agnostic model, we no longer assume a function $f \in \mathscr{F}$ that predicts the log-cost y perfectly. It is possible that the true predictor is outside $\mathscr{F}$, or in more difficult scenarios for any feature x, the behaviour of the log-cost y may be entirely arbitrary.

We first show that the loss function ε-parameterized competitive error defined earlier is still a reasonable proxy for the competitive ratio. Specifically, we show that any algorithm that hopes to achieve a competitive ratio of $1 + O(\varepsilon)$ must use a prediction $\hat{f} \in \mathscr{F}$ whose error $\mathbf{ER}_{\mathbb{D},\varepsilon}(f)$ is bounded by $O(\varepsilon)$. We formally state this below:

Lemma 20. *Let $\mathscr{A}$ be an algorithm for* LEARNTOSEARCH *that has access to a predictor $\hat{f} : \mathbb{X} \mapsto [0, H]$ for the log-cost y. Then, there exists a distribution $\mathbb{D}$ and a function $\hat{f}_{\mathscr{A}}$ with the property $\mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f}_{\mathscr{A}}) = \varepsilon$ such that $\mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\mathrm{CR}_{\mathscr{A}}(x,y)\right] \geqslant 1 + \frac{\varepsilon}{2}$.*

Proof. Let the predicted log-cost be $\hat{y} = f_{\mathscr{A}}(x)$. Let $\phi(\hat{y})$ be the sum total of the costs of solutions bought by $\mathscr{A}$ till the optimal log-cost reaches $\hat{y}$. Clearly $\phi(\hat{y}) \geqslant e^{\hat{y}}$. Since the

algorithm $\mathscr{A}$ can be possibly randomized, let $e^{\hat{y}} \leqslant \phi(\hat{y}) \leqslant e^{\hat{y}+\varepsilon}$ with probability α over the distribution chosen by $\mathscr{A}$.

We define the distribution $\mathbb{D}$ as: $\mathbb{X}$ is just the singleton set $\{x_0\}$ and $\mathbb{Y} = \{\hat{y}, \hat{y}\cdot(1+\varepsilon)\}$. The distribution $\mathbb{D}$ assigns probability $1-\varepsilon$ to $(x_0, \hat{y})$ and ε to $(x_0, \hat{y}\cdot(1+\varepsilon))$ (note that the optimal cost is $e^{\hat{y}}$ and $e^{\hat{y}\cdot(1+\varepsilon)}$ in these cases respectively). Note that $\mathbb{D}$ and $f_{\mathscr{A}}$ satisfy $\mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f}_{\mathscr{A}}) = \varepsilon$. The expected competitive ratio of $\mathscr{A}$ is a least

$$\text{CR} \geqslant (2\alpha+1-\alpha)\cdot\varepsilon + \alpha\cdot(1-\varepsilon) + (1-\varepsilon)\cdot(1-\alpha)\cdot(1+\varepsilon) = 1+\varepsilon-(1-\alpha)\cdot\varepsilon^2 \geqslant 1+\frac{\varepsilon}{2}.$$

□

Unlike in the standard model, we no longer have that for any $\varepsilon > 0$, $\min_{f\in\mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f) = 0$. Therefore, we need to first quantify the performance of an *ideal* algorithm that uses predictors from $\mathscr{F}$.

Definition 19. *Let $\chi(\varepsilon) = \min_{f\in\mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f)$. Then, $\Delta_{\mathscr{F}}$ is the solution to the equation: $\varepsilon = \chi(\varepsilon)$.*

$\Delta_{\mathscr{F}}$ measures the best competitive ratio that we can hope to get when we use a predictor from $\mathscr{F}$. Note that ε appears in two places in this definition, since the loss function in Definition 14 depends on ε. We first show that this is a reasonable definition in that the solution to the equation is unique:

Lemma 21. *For a given function family $\mathscr{F}$ and distribution $\mathbb{D}$, the value of $\Delta_{\mathscr{F}}$ is unique.*

Proof of Lemma 21. Let $\chi(\varepsilon) = \min_{f\in\mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f)$. Note that, $\chi(\varepsilon)$ is non-increasing in ε, and $\lim_{\varepsilon\to 0}\chi(\varepsilon) > 0$. Since $\ell_\varepsilon(\cdot,\cdot) \leqslant \frac{5}{\varepsilon} - 1$, we have $\chi(2) < 2$. Therefore, there must exist $\Delta_{\mathscr{F}} \in (0,2)$ such that:

$$\Delta_{\mathscr{F}} = \min_{f\in\mathscr{F}} \mathbf{ER}_{\mathbb{D},\Delta_{\mathscr{F}}}(f)$$

The uniqueness follows from the monotonicity of the function $\chi(.)$ □

We also give an algorithm that can approximate $\Delta_{\mathscr{F}}$ (Algorithm 10).

Algorithm 10 Procedure to estimate $\Delta_{\mathscr{F}}$

Input: Sample Set S, and function family $\mathscr{F}$
Let ε be an accuracy parameter given by the size of the sample set S.
Choose $\varepsilon := \varepsilon$
Compute: $\hat{f}$ such that $\mathbf{ER}_{S,\varepsilon}(\hat{f}) \leqslant \min_{f\in\mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f) + \frac{\varepsilon}{3}$.
while $\varepsilon \leqslant \mathbf{ER}_{S,\varepsilon}(\hat{f})$
  $\varepsilon \leftarrow 2\varepsilon$.
  Recompute $\hat{f}$ s.t. $\mathbf{ER}_{S,\varepsilon}(\hat{f}) \leqslant \min_{f\in\mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f) + \frac{\varepsilon}{3}$.
Return ε.

Lemma 22. *If* $|S| \geqslant C \cdot \left(\frac{H\cdot d \log\frac{1}{\varepsilon}\log\frac{1}{\delta}}{\varepsilon}\right)$ *for suitable constants* $C > 0, \delta \leqslant \frac{1}{2}$, *and* $\varepsilon \leqslant \Delta_{\mathscr{F}}$, *then with probability at least* $1-\delta$, *we have* $\frac{5}{36}\cdot\varepsilon \leqslant \Delta_F \leqslant \frac{17}{8}\varepsilon$, *where* ε *is as returned by Algorithm 10.*

Proof of Lemma 22. Let $\chi(\varepsilon) = \min_{f\in\mathscr{F}} \mathbf{ER}_{\mathbb{D},\varepsilon}(f)$ and $\lambda(\varepsilon) = \min_{f\in\mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f)$, where $S \sim \mathbb{D}^m$. For fixed $\hat{y}, y$, we note that $\ell_\varepsilon(y,\hat{y})$ can only decrease when ε increases. Therefore, both $\chi(\varepsilon)$ and $\lambda(\varepsilon)$ are non-increasing with ε.

From Lemma 11, we have : $\mathscr{N}\left(\frac{\varepsilon^{\frac{3}{2}}}{8}, \mathscr{F}, 2m\right) \leqslant \left(\frac{1}{\varepsilon}\right)^{O(d)}$. Noting that the size of the sample set m exceeds $C\cdot\left(\frac{H\cdot d\cdot\log\frac{1}{\varepsilon}\cdot\log\frac{1}{\delta}}{\varepsilon}\right)$ for some large $C \geqslant 0$, we use Lemma 12 with $\eta^2 = \frac{\varepsilon}{16}$ and $\alpha = 1$ to claim that with probability $1-\delta$, we have for all $f \in \mathscr{F}$:

$$\mathbf{ER}_{S,\varepsilon}(f) \leqslant \frac{3}{2}\cdot\mathbf{ER}_{\mathbb{D},\varepsilon}(f) + \frac{\varepsilon}{8}. \tag{3.2}$$

and,

$$\mathbf{ER}_{\mathbb{D},\varepsilon}(f) \leqslant 2\cdot\mathbf{ER}_{S,\varepsilon}(f) + \frac{\varepsilon}{8}. \tag{3.3}$$

Due to the breaking condition, we have $\varepsilon \geqslant \mathbf{ER}_{S,\varepsilon}(\hat{f}) \geqslant \lambda(\varepsilon)$. Then, by Eq. (3.3), we

have:

$$\chi(\varepsilon) \leqslant 2 \cdot \lambda(\varepsilon) + \frac{\varepsilon}{8} \leqslant \frac{17}{8} \cdot \varepsilon.$$

By monotonicity of $\chi(.)$,

$$\chi\left(\frac{17}{8}\varepsilon\right) \leqslant \chi(\varepsilon) \leqslant \frac{17}{8} \cdot \varepsilon. \tag{3.4}$$

Also, we have that $\frac{\varepsilon}{2} < \lambda(\frac{\varepsilon}{2}) + \frac{\varepsilon}{6}$. Let $f^* = \arg\min_{f \in \mathscr{F}} \mathbf{ER}_{\mathbb{D}, \frac{\varepsilon}{2}}(f)$, then using Eq. (3.2) on f^*, we get:

$$\begin{aligned}\lambda\left(\frac{\varepsilon}{2}\right) &\leqslant \mathbf{ER}_{S, \frac{\varepsilon}{2}}(f^*) \\ &\leqslant \frac{3}{2} \cdot \chi\left(\frac{\varepsilon}{2}\right) + \frac{\varepsilon}{8}\end{aligned}$$

Hence,

$$\frac{5\varepsilon}{36} \leqslant \chi\left(\frac{\varepsilon}{2}\right) \leqslant \chi\left(\frac{5\varepsilon}{36}\right).$$

Combining the above with Eq. (3.4), we get $\frac{5}{36} \cdot \varepsilon \leqslant \Delta_F \leqslant \frac{17}{8}\varepsilon$. □

We are now ready to define our LEARNTOSEARCH algorithm for the agnostic model. This algorithm is simply Algorithm 9 where the accuracy parameter ε is set to the value of ε returned by Algorithm 10.

Theorem 21. *In the agnostic model for a function family $\mathscr{F}$, Algorithm 9 with accuracy parameter ε from Algorithm 10 obtains a competitive ratio of $1 + O(\Delta_{\mathscr{F}})$ with probability at least $1 - \delta$, when using $O\left(\frac{H \cdot d \log\left(\frac{1}{\Delta_{\mathscr{F}}}\right) \cdot \log\frac{1}{\delta}}{\Delta_{\mathscr{F}}}\right)$ samples, where $d = Pdim(\mathscr{F})$.*

Proof. From the breaking condition in Algorithm 10 and Lemma 22, we have that $\arg\min_{f \in \mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f) \leqslant \varepsilon \leqslant \frac{36}{5} \cdot \Delta_{\mathscr{F}}$. Using the sample error minimization algorithm returns a function $\hat{f}$ such that

$$\mathbf{ER}_{S,\varepsilon}(\hat{f}) \leqslant \min_{f \in \mathscr{F}} \mathbf{ER}_{S,\varepsilon}(f) + \varepsilon \leqslant 2\varepsilon$$

Finally, application of Lemma 12 to bound $\mathbf{ER}_{\mathbb{D},\varepsilon}(\hat{f}) = O(\Delta_{\mathscr{F}})$, followed by Lemma 10 gives the desired result. □

3.6.4 Lower Bound construction

We also lower bound the sample complexity of a LEARNTOSEARCH algorithm:

Theorem 22. *Any* LEARNTOSEARCH *algorithm that is ε-efficient with probability at least $1-\delta$ must query $\Omega\left(\frac{\log\frac{1}{\delta}}{\varepsilon^2}\right)$ samples.*

We begin with the agnostic case. We describe a class of distributions $\mathbb{D}_p$ on pairs (x,y), where p is a parameter in $(0,1)$. Recall that y represents $\log_2 z$, where z is the actual optimal cost of the offline-instance. The distribution $\mathbb{D}_p$ consists of two pairs: $(1,1)$ with probability p, and $(0,2)$ with probability $1-p$. Note that the projection of $\mathbb{D}_p$ on the first coordinate is a Bernoulli random variable with probability of 1 being p. For the sake of concreteness, the input sequence $\Sigma = \tau_0, \tau_1, \ldots,$ is such that $\textsc{opt}(0) = 2, \textsc{opt}(1) = 4$. The distribution $\mathbb{D}_p$ ensures that the stopping time parameter $T = 0$ with probability p, and $T = 1$ with probability $1-p$. It follows that any online algorithm has only one decision to make: whether to buy the solution for $\mathscr{I}_0$.

Let $\mathscr{A}_p^\star$ be the algorithm which achieves the minimum competitive ratio when the input distribution is $\mathbb{D}_p$, and let $\rho_p^\star$ be the expected competitive ratio of this algorithm. There are only two strategies for any algorithm: (i) buy optimal solution for $\mathscr{I}_0$, and if needed buy the solution for $\mathscr{I}_1$, or (ii) buy the optimal solution for $\mathscr{I}_1$ at the beginning. The following result determines the value of $\rho_p^\star$.

Lemma 23. *If $p = \frac{1}{3} + \varepsilon$ for some $\varepsilon \geqslant 0$, then $\rho_p^\star = \frac{4}{3} - \frac{\varepsilon}{2}$, and strategy (i) is optimal here. In case $p = \frac{1}{3} - \varepsilon$ for some $\varepsilon \geqslant 0$, then $\rho_p^\star = \frac{4}{3} - \varepsilon$, and strategy (ii) is optimal.*

Proof. For strategy (i), the cost of the algorithm is 2 with probability p and 6 with proba-

bility $1-p$. Therefore its expected competitive ratio is

$$p \cdot 1 + \frac{6}{4} \cdot (1-p) = \frac{3}{2} - \frac{p}{2}.$$

For strategy (ii), the cost of the algorithm is always 4. Therefore, its expected competitive ratio is

$$p \cdot 2 + 1 \cdot (1-p) = p + 1.$$

It follows that strategy (i) is optimal when $p \geqslant 1/3$, whereas strategy (ii) is optimal when $p \leqslant 1/3$. □

We are now ready to prove Theorem 22. Let $\mathscr{A}$ be an algorithm for LTS which is $\varepsilon/4$-efficient with probability at least $1-\delta$. Further, let k be an upper bound on the sample complexity of $\mathscr{A}$. Given k samples from a distribution $\mathbb{D}_p$, the algorithm outputs a strategy which is a probability distribution on strategies (i) and (ii). We use this algorithm $\mathscr{A}$ to solve the following prediction problem $\mathscr{P}$: X is a random variable uniformly distributed over $\{\frac{1}{3}-\varepsilon, \frac{1}{3}+\varepsilon\}$. Given i.i.d. samples from from 0-1 Bernoulli random variable T with probability of 1 being X, we would like to predict the value of X.

Lemma 24. *Let $\mathscr{A}$ be an algorithm for LTS which is $\varepsilon/4$-efficient with probability at least $1-\delta$. Then, there is an algorithm that predicts the value of X with probability at least $1-\delta$ using k i.i.d. samples from T.*

Proof. Let $t_1, \ldots, t_k$ be i.i.d. samples from T. We give k samples $(x_1, y_1), \ldots, (x_k, y_k)$ to $\mathscr{A}$ as follows: for each $i = 1, \ldots, k$, if $t_i = 0$, we set (x_i, y_i) to $(0,2)$; else we set it to $(1,1)$. Observe that the samples given to $\mathscr{A}$ are k i.i.d. from the distribution $\mathbb{D}_X$.

Based on these samples, $\mathscr{A}$ puts probability q_1 on strategy (i) (and $1-q_1$ on strategy (ii)). If $q_1 > 1/2$, we predict $X = \frac{1}{3}+\varepsilon$, else we predict $X = \frac{1}{3}-\varepsilon$.

We claim that this prediction strategy predicts X correctly with probability at least $1-\delta$. To see this, assume that $\mathscr{A}$ is $\varepsilon/4$-efficient (which happens with probability at least $1-\delta$).

First consider the case when $X = \frac{1}{3} + \varepsilon$. In this case, Lemma 23 shows that the expected competitive ratio of $\mathscr{A}$ is at most $\frac{4}{3} - \frac{\varepsilon}{4}$. As in the proof of Lemma 23, the expected competitive ratio of $\mathscr{A}$ is

$$q_1 \left(\frac{3}{2} - \frac{X}{2} \right) + (1 - q_1)(X + 1).$$

We argue that $q_1 \geqslant 1/2$. Suppose not. Since $X > 1/3$, $\frac{3}{2} - \frac{X}{2} \leqslant X + 1$. Therefore, the above is at least (using $q_1 \leqslant 1/2$ and $X = 1/3 + \varepsilon$)

$$\frac{1}{2} \left(\frac{3}{2} - \frac{X}{2} \right) + \frac{1}{2}(X + 1) > 4/3,$$

which is a contradiction. Therefore $q_1 > 1/2$.

Now consider the case when $X = 1/3 - \varepsilon$. Again Lemma 23 shows that the expected competitive ratio of $\mathscr{A}$ is at most $\frac{4}{3} - \frac{3\varepsilon}{4}$. It is easy to check that if $q_1 \geqslant 1/2$, then the expected competitive ratio of $\mathscr{A}$ is at least

$$\frac{4}{3} - \frac{\varepsilon}{4},$$

which is a contradiction. Therefore, $q_1 < 1/2$. This proves the desired result. □

It is well known that in order to predict X with probability at least $1 - \delta$, we need $\Omega\left(\frac{1}{\varepsilon^2} \ln\left(\frac{1}{\delta} \right) \right)$ samples. This proves Theorem 22.

3.6.5 Robustness of Algorithm 9

So far, we have established the competitive ratio of Algorithm 9 in the PAC model. Now, we show the robustness of this algorithm, i.e., bound its competitive ratio for *any* input. Even for adversarial inputs, we show that this algorithm has a competitive ratio of $O(1/\varepsilon)$, which matches the robustness guarantees in Theorem 17 for the PREDICT-AND-DOUBLE algorithm.

Theorem 23. *Algorithm 9 is* $5(1+\frac{1}{\varepsilon}) = O\left(\frac{1}{\varepsilon}\right)$*-robust.*

Proof of Theorem 23. We show this theorem by using the following lemma:

Lemma 25. *Let* $\mathscr{A}$ *denote Algorithm 9. Then*

$$\text{CR}_{\mathscr{A}}(x,y) = \begin{cases} 4 \text{ when } y \leqslant \ln\frac{5}{\varepsilon} - \hat{y} \text{ or } y > \hat{y} + \ln\left(1+\frac{\varepsilon}{5}\right) \\ (1+\varepsilon)\cdot e^{\hat{y}-y} \text{ otherwise} \end{cases}$$

Proof. The proof follows from Lemma 9 by noting that $e^y = \text{OPT}(T), e^{\hat{y}} = \text{OPT}(\hat{T})$, and $\text{OPT}(t_1) = \frac{\varepsilon}{5}\cdot e^{\hat{y}}, \text{OPT}(t_2) = (1+\frac{\varepsilon}{5})\cdot e^{\hat{y}}$. □

Theorem 23 now follows by noting that the worst case is when y just exceeds t_1, i.e., when $y = \ln\frac{5}{\varepsilon} - \hat{y}$. □

3.7 Inadequacy of Traditional Loss Functions

In this work, we explore the idea of a carefully crafted loss function that can help in making better predictions for the online decision task. To illustrate this, consider the problem of balancing the load between machines/clusters in a data center where remote users are submitting jobs. The goal is to minimize the maximum load on any machine, also called the makespan of the assignment. The optimal makespan, which we would like to predict, depends on the workload submitted by individual users who are currently active in the system. Therefore, we would like to use the user features to predict their behavior in terms of the workload submitted to the server. A typical feature vector would then be a binary vector encoding the set of users currently active in the system, and based on this information, a learning model trained on historical behavior of the users can predict (say) a histogram of loads that these users are expected to submit, and therefore, the value of the optimal makespan. The feature space can be richer, e.g., including contextual information like the time of the day, day of the week, etc. that are useful to more accurately predict user behavior. Irrespective of the precise learning model, the main idea is that the learner

should try to optimize for competitive loss instead of standard loss functions. This is because the goal of the learner is not to accurately predict the workload of each user, but to eventually obtain the best possible makespan. For instance, a user who submits few jobs that are inconsequential to the eventual makespan need not be accurately predicted. Our technique automatically makes this adjustment in the loss function, thereby obtaining better performance on the competitive ratio.

In this section, we motivate the use of asymmetric loss function (Definition 14) by showing that an algorithm which uses predictions from a learner minimizing a symmetric loss function, such as absolute loss or squared loss, would have a large competitive ratio. The intuition is that if we err on either side of the true value of T by the same amount, the competitive ratio in the two cases does not scale in the same manner. To formalize this intuition, we define a class of distributions $\mathbb{D}_\Delta$, parameterized by $\Delta > 0$, which are symmetric around a real c; more concretely this distribution places equal weight on $\{c-\Delta, c+\Delta\}$. Any algorithm relying on a symmetric loss function would always predict c. In such a case, the online algorithm $\mathscr{A}$ has no new information. However, if Δ is large, an offline algorithm is better off buying the solution till $c-\Delta$ first, whereas if Δ is small, it should buy the solution for $c+\Delta$ in the first step. An algorithm which relies only on c would err in one of these two cases. This idea is formalized in Lemma 26. Our second result (Lemma 27) shows that predicting log-loss within an additive ε factor may result in a $1+\Omega\left(\sqrt{\varepsilon}\right)$ expected competitive ratio. This further bolsters the case for the loss function as in Definition 14.

Lemma 26. *Let $\mathscr{A}$ be an algorithm that uses predictions made by a learner that minimizes symmetric error. Then, one of the following statements is true:*

1. $\mathbb{E}_{(x,y)\sim\mathbb{D}_\Delta}\left[\mathrm{CR}_{\mathscr{A}}(x,y)\right]$ *is* $\Omega(e^\Delta)$ *when* $\Delta \geqslant 4$.

2. $\mathbb{E}_{(x,y)\sim\mathbb{D}_\Delta}\left[\mathrm{CR}_{\mathscr{A}}(x,y)\right] \geqslant 1+\Omega(1)$, *when* $\Delta = \varepsilon$, *with* ε *being an arbitrarily small positive real number.*

Proof. We define a family of distributions $\mathbb{D}_\Delta$, parameterized by Δ, $0 \leqslant \Delta \leqslant c$, where c is a large enough constant, as follows:

Definition 20. *Let $\mathbb{X}$ denote the singleton set $\{x_0\}$ and $\mathbb{Y}$ denote $\{c-\Delta, c+\Delta\}$. The distribution $\mathbb{D}_\Delta$ on $\mathbb{X} \times \mathbb{Y}$ assigns probability $\frac{1}{2}$ to both $(x_0, c-\Delta)$ and $(x_0, c+\Delta)$.*

Ideally, we would want an algorithm $\mathscr{A}$ to satisfy $\mathbb{E}_{(x,y)\sim\mathbb{D}_\Delta}[\mathrm{CR}_{\mathscr{A}}(x,y)] \to 0$ when $\Delta \to 0$, while still maintaining a worst-case result like $\mathbb{E}_{(x,y)\sim\mathbb{D}_\Delta}[\mathrm{CR}_{\mathscr{A}}(x,y)] \leqslant O(1)$. The following construction shows that using a symmetric loss function would not be helpful. Suppose we use a learner that outputs the function which minimizes a symmetric loss function. Then given samples from $\mathbb{D}_\Delta$, such a learner will always yield $\hat{y} = c$ as the prediction.

Since the feature is fixed, the behavior of the algorithm is independent of the feature and hence, it only needs to decide on a list of solutions that it will progressively buy. Let τ be the cost of the first solution bought by $\mathscr{A}$ that lies inside the interval $[e^{c-\Delta}, e^{c+\Delta}]$ where $\hat{y} = c$ is the predicted log-cost that has been supplied to the algorithm.

There are two possible cases for τ:

(i) $\tau \geqslant e^c$: With probability $\frac{1}{2}$, the competitive ratio is $\frac{e^c}{e^{c-\Delta}} = e^\Delta$. Hence, $\mathbb{E}[\mathrm{CR}_{\mathscr{A}}(x,y)] \geqslant \frac{1+e^\Delta}{2}$. Observe that if $\Delta \geqslant 4$, then $\mathbb{E}[\mathrm{CR}_{\mathscr{A}}(x,y)]$ is $\Omega(e^\Delta)$, and hence unbounded.

(ii) $\tau < e^c$: With probability $\frac{1}{2}$, $y = c+\Delta$, in which case the competitive ratio is $\frac{e^c+e^{c+\Delta}}{e^{c+\Delta}} = 1+e^{-\Delta}$. Therefore, $\mathbb{E}[\mathrm{CR}_{\mathscr{A}}(x,y)]$ is $1+\frac{e^{-\Delta}}{2} = 1+\Omega(1)$, even when Δ is an arbitrarily small positive ε. □

It is worth noting that if we use the loss function as in Definition 14, then Algorithm 9 has expected competitive ratio $1+O(\varepsilon)$ when $\Delta \leqslant \varepsilon$. Further, this algorithm defaults to DOUBLE when $\Delta = \Omega(1)$, and hence has bounded competitive ratio in this case.

We also show that any algorithm which relies on a predictor of log-cost which has an ε bound on the absolute loss must incur $1+\Omega(\sqrt{\varepsilon})$ expected competitive ratio. Comparing

this result with Theorem 21 shows that our loss function defined as in Definition 14 gives better competitive ratio guarantees.

Lemma 27. *Let $\mathscr{A}$ be a learning-augmented algorithm for* ONLINESEARCH*, that has access to a predictor $P : \mathbb{X} \mapsto [0,H]$ that predicts the log-cost y. Moreover, the only guarantee on P is that $\mathbb{E}_{(x,y)\sim\mathbb{D}}[|P(x)-y|] \leqslant \varepsilon$. Then there is a distribution $\mathbb{D}$ and a predictor P such that $\mathbb{E}_{(x,y)\sim\mathbb{D}}[\mathrm{CR}_{\mathscr{A}}(x,y)] \geqslant 1+\frac{\sqrt{\varepsilon}}{2}$.*

Proof. Fix an algorithm $\mathscr{A}$. Given the prediction $\hat{y} = 1$, the algorithm outputs a (randomized) strategy for buying optimal solutions at several time steps. Let ϕ be the sum total of the costs of the solutions bought by the algorithm before the cost of the optimal solution reaches e. Clearly $\phi \geqslant e$. Let α denote the probability that $\phi \in [e, e^{1+\sqrt{\varepsilon}}]$, where the probability is over the distribution chosen by $\mathscr{A}$.

We define the distribution $\mathbb{D}$ as follows: $\mathbb{X}$ is just the singleton set $\{x_0\}$ and $\mathbb{Y} = \{1, 1+\sqrt{\varepsilon}\}$. The distribution $\mathbb{D}$ assigns probability $1-\sqrt{\varepsilon}$ to $(x_0, 1)$ and $\sqrt{\varepsilon}$ to $(x_0, 1+\sqrt{\varepsilon})$ (note that the optimal cost is e and $e^{1+\sqrt{\varepsilon}}$ in these cases respectively). Note that $\mathbb{E}_{\mathbb{D}}[y]$ is $1+\varepsilon$. Consider the predictor P which outputs the prediction $\hat{y} = 1$, and therefore satisfies the condition $\mathbb{E}_{(x,y)\sim\mathbb{D}}[|P(x)-y|] \leqslant \varepsilon$. The expected competitive ratio of $\mathscr{A}$ is a least

$$(1-\sqrt{\varepsilon})\left[\alpha\cdot 1 + (1-\alpha)\cdot e^{\sqrt{\varepsilon}}\right] +$$

$$\sqrt{\varepsilon}\left[\alpha\frac{e+e^{1+\sqrt{\varepsilon}}}{e^{1+\sqrt{\varepsilon}}} + (1-\alpha)\cdot 1\right]$$

Approximating $e^{\sqrt{\varepsilon}}$ by $1+\sqrt{\varepsilon}$, the above expression simplifies to

$$1+\sqrt{\varepsilon}-\varepsilon \geqslant 1+\frac{\sqrt{\varepsilon}}{2}.$$

□

3.8 Conclusion

In this chapter, we studied the role of regression in making predictions for learning-augmented online algorithms. In particular, we used the ONLINESEARCH framework that includes a variety of online problems such as ski rental and its generalizations, online scheduling, online bin packing, etc. and showed that by using a carefully crafted loss function, we can obtain predictions that yield near-optimal algorithms for this problem. One assumption that holds for the above problems, but not for other problems such as online matching, is the composability of solutions, i.e., that the union of two feasible solutions is also a feasible solution. Extending our work to such "packing" problems is an interesting direction for future research. Another interesting direction would be to give a general recipe for converting competitive ratios to loss functions, minimizing which over a collection of training samples generates better ML predictions for online problems.

3.9 Chapter Notes

The work presented in this chapter was done jointly with Rong Ge, Amit Kumar and Debmalya Panigrahi. A preliminary version appeared in the Advances in Neural Information Processing Systems 34 (NeurIPS 2021) (see Anand et al. (2021))

4

Online Algorithms with Multiple Predictions

In this chapter, we give a generic algorithmic framework for online covering problems with multiple predictions that obtains an online solution that is competitive against the performance of the best solution obtained from the predictions.

4.1 Introduction

In this chapter, we focus on online algorithms with *multiple* machine-learned predictions. In many situations, different ML models and techniques end up with distinct predictions about the future, and the online algorithm has to decide which prediction to use among them. Indeed, this is also true of human experts providing inputs about expectations of the future, or other statistical tools for predictions such as surveys, polls, etc. Online algorithms with multiple predictions were introduced by Gollapudi and Panigrahi Gollapudi and Panigrahi (2019b) for the ski rental problem, and has since been studied for multi-shop ski rental Wang et al. (2020) and facility location Almanza et al. (2021). Furthermore, Bhaskara et al. (2020) considers multiple hints for regret minimization in Online Linear Optimization. In this chapter, instead of focusing on a single problem, we extend the

powerful paradigm of *online covering problems* to incorporate multiple predictions. As a consequence, we obtain online algorithms with multiple predictions for a broad range of classical problems such as *set cover*, *caching*, and *facility location* as corollaries of the general technique that we develop here.

The Online Covering Framework. Online covering is a powerful framework for capturing a broad range of problems in combinatorial optimization. In each online step, a new linear constraint $\mathbf{a} \cdot \mathbf{x} \geqslant b$ is presented to the algorithm, where $\mathbf{x}$ is the vector of variables, $\mathbf{a}$ is a vector of non-negative coefficients, and b is a scalar. The algorithm needs to satisfy the new constraint, and is only allowed to increase the values of the variables to do so. The goal is to minimize an objective function $\mathbf{c} \cdot \mathbf{x}$, where $\mathbf{c}$ is the vector of costs that is known offline. This formulation captures a broad variety of problems including set cover, (weighted) caching, revenue maximization, network design, ski rental, TCP acknowledgment, etc. Alon *et al.* Alon et al. (2009) proposed a multiplicative weights update (MWU) technique for this problem and used it to solve the online set cover problem. This was quickly adapted to other online covering problems including weighted caching Bansal et al. (2007), network design Alon et al. (2006), allocation problems for revenue maximization Buchbinder et al. (2007), etc. (The reader is referred to the survey Buchbinder and Naor (2009a) for more examples.) All these algorithms share a generic method for obtaining a fractional solution to the online covering problem, which was formalized by Buchbinder and Naor Buchbinder and Naor (2009b) and later further refined by Gupta and Nagarajan Gupta and Nagarajan (2014). Since then, the online covering problem has been generalized to many settings such as convex (non-linear) objectives Azar et al. (2016) and mixed covering and packing problems Azar et al. (2013).

Comparison with Prior Work on Online Covering with ML Prediction. Bamas, Maggiori, and Svensson Bamas et al. (2020) were the first to consider the online covering framework in

the context of ML predictions. In a beautiful work, they gave the first general-purpose tool for online algorithms with predictions, and showed that this can be used to solve several classical problems like set cover and dynamic TCP acknowledgment. In their setting, a solution is presented as advice to the online algorithm at the outset, and the algorithm incorporates this suggestion in its online decision making.

We give a general scheme for the online covering framework with *multiple* predictions. In particular:

- Since we are in the multiple predictions setting, we allow $k > 1$ suggestions instead of just a single suggestion, and benchmark our algorithm's performance against the best suggested solution. (Of course, the best suggestion is not known to the algorithm.)

- In contrast to Bamas *et al.* Bamas et al. (2020), we do not make the assumption that the entire suggested solution is given up front. Instead, in each online step, each of the k suggestions gives a feasible way of satisfying the new constraint. Note that this is more general than giving the suggested solution(s) up front, since the entire solution(s) can be presented in each online step as a feasible way of satisfying the new constraint.

- In terms of the analysis, we extend the potential method from online algorithms (in contrast, Bamas *et al.* Bamas et al. (2020) use the primal dual framework). The potential method has been used recently for many classic problems in online algorithms such as weighted paging Bansal et al. (2010), k-server Buchbinder et al. (2019), metric allocation Bansal and Coester (2021), online set cover Buchbinder et al. (2019), etc. In fact, it can also be used to reprove the main results of Bamas *et al.* Bamas et al. (2020) in the single prediction setting. In this chapter, we extend this powerful tool to incorporate multiple ML predictions.

- Finally, we show that our techniques extend to a generalization of the online covering

framework to include *box*-type constraints. This extension allows the framework to handle more problems such as online facility location that are not directly captured by the online covering framework.

Comparison with Online Learning. The reader will notice the similarity of our problem to the classical experts' framework from online learning (see, e.g., the survey Shalev-Shwartz (2012)). In the experts' framework, each of k experts provides a suggestion in each online step, and the algorithm has to choose (play) one of these k options. After the algorithm has made its choice, the cost (loss) of each suggestion is revealed before the next online step. The goal of the algorithm is to be competitive with the *best expert* in terms of total loss. In contrast,

- Since we are solving a combinatorial problem, the (incremental) cost of any given step for an expert or the algorithm depends on their pre-existing solution from previous steps (therefore, in particular, even after following an expert's choice, the algorithm might suffer a larger incremental cost than the expert). This is unlike online learning where the cost in a particular step is independent of previous choices.

- In online learning, the algorithm is benchmarked against the best *static* expert in hindsight, i.e., the best solution whose choices match that of the same expert across all the steps. Indeed, it can be easily shown that no algorithm can be competitive against a *dynamic* expert, namely a solution that chooses the best suggestion in each online step even if those choices come from different experts. Observe that such a dynamic expert can in general perform much better than each of the suggestions, e.g., when the suggestions differ from each other but at each time, at least one of them suggests a good solution. But, in our problem, since the choices made by experts correspond to solutions of a combinatorial problem, we can actually show that our algorithm is competitive even against a dynamic expert. Namely, the k suggestions

in every step are not indexed by specific experts, and the algorithm is competitive against any composite solution that chooses any one of the k suggestions in each step.

– In online learning, the goal is to obtain *regret* bounds that show that the online algorithm approaches the best (static) expert's performance up to additive terms. Such additive guarantees are easily ruled out for our problem, even for a static expert. As is typically the case in online algorithms, our performance guarantees are in the form of (multiplicative) competitive ratios rather than (additive) regret bounds.

4.1.1 Our Contributions.

Our first contribution is to formalize the online covering problem with multiple predictions (Section 4.2). Recall that in each online step, along with a new constraint, the algorithm receives k feasible suggestions for satisfying the constraint. Using these suggestions, we design an algorithm for obtaining a fractional solution to the online covering problem–that we call the OCP algorithm (Section 4.3). To compare this algorithm to the best suggested solution, we define a benchmark DYNAMIC that captures the minimum cost (fractional) solution that is consistent with at least one of the suggestions in each online step.

– Our main technical result shows that the cost of the solution produced by the OCP algorithm is at most $O(\log k)$ times that of the DYNAMIC solution.

It is noteworthy that unlike in the classical online covering problem (without predictions), the competitive ratio is independent of the problem size, and only depends on the number of suggestions k. As the number of suggestions increases, the competitive ratio degrades because the suggestions have higher entropy (i.e., are less specific). As two extreme examples, consider $k = 1$, in which case it is trivial for an algorithm to exactly match the DYNAMIC benchmark simply by following the suggestion in each step. In contrast, when k is very large, the set of suggested solutions can essentially include all possible solutions, and therefore, the suggestions are useless.

The analysis of the OCP algorithm makes careful use of potential functions that might be of independent interest. But, while the analysis of the OCP algorithm is somewhat intricate, we note that the algorithm itself is extremely simple.

- We show that the competitive ratio of $O(\log k)$ obtained by the OCP algorithm is tight. We give a lower bound of $\Omega(\log k)$ by only using binary (0/1) coefficients and unit cost for each variable, which implies that the lower bound holds even for the special case of the unweighted set cover problem.

- Using standard techniques, we observe that the OCP algorithm can be *robustified*, i.e., along with being $O(\log k)$-competitive against the best suggested solution, the algorithm can be made $O(\alpha)$-competitive against the optimal solution where α is the competitive ratio of the best online algorithm (without predictions).

We then use the OCP algorithm to solve two classic problems–online **set cover** (Section 4.4) and **caching** (Section 4.5)–in the multiple predictions setting.

- We generalize the online covering framework by introducing *box*-type constraints (Section 4.6). We show that our techniques and results from online covering extend to this more general setting.

We then use this more general formulation for solving the classical online **facility location** problem (Section 4.7).

4.2 The Online Covering Framework

4.2.1 Problem Statement

We define the *online covering problem* (OCP) as follows. There are n non-negative variables $\{x_i : i \in [n]\}$ where each $x_i \in [0,1]$. Initially, $x_i = 0$ for all $i \in [n]$. A linear objective function $c(x) := \sum_{i=1}^{n} c_i x_i$ is also given offline. In each online step, a new *covering* constraint is

presented, the j-th constraint being given by $\sum_i a_{ij}x_i \geqslant 1$ where $a_{ij} \geqslant 0$ for all $i \in [n]$.[1] The algorithm is only allowed to *increase* the values of the variables, and has to satisfy the new constraint when it is presented. (We denote the total number of constraints by m.) The goal is to minimize the objective function $c(x)$. We write this succinctly below:

$$\min_{x_i \in [0,1]: i \in [n]} \left\{ \sum_{i=1}^{n} c_i x_i : \sum_{i=1}^{n} a_{ij} x_i \geqslant 1 \ \forall j \in [m] \right\}.$$

This framework captures a large class of algorithmic problems such as (fractional) set cover, caching, etc. that have been extensively studied in the online algorithms literature. Our goal will be to obtain a generic algorithm for OCP with multiple suggestions. When the j-th constraint is presented online, the algorithm also receives k suggestions of how the constraint can be satisfied. We denote the s-th suggestion for the j-th constraint by variables $x_i(j,s)$; they satisfy $\sum_{i=1}^{n} a_{ij} x_i(j,s) \geqslant 1$, i.e., all suggestions are feasible.

To formally define the *best* suggestion, we say that a solution $\{x_i : i \in [n]\}$ is *supported* by the suggestions $\{x_i(j) : i \in [n], j \in [m]\}$ if $x_i \geqslant x_i(j)$ for all $j \in [m]$. Using this definition, we consider below two natural notions of the best suggestion that we respectively call the *experts setting* and the *multiple predictions setting*.

The Experts Setting. In the experts setting, there are k experts, and the s-th suggestion for each constraint comes from the same fixed expert $s \in [k]$ (say some fixed ML algorithm or a human expert). The online algorithm is required to be competitive with the *best* among these k experts[2]. To formalize this, we define the benchmark:

$$\text{STATIC} = \min_{s \in [k]} \sum_{i=1}^{n} c_i \cdot \max_{j \in [m]} x_i(j,s).$$

[1] A more general definition allows constraints of the form $\sum_{i=1}^{n} a_{ij} x_i \geqslant b_j$ for any $b_j > 0$, but restricting b_j to 1 is without loss of generality since we can divide throughout by b_j without changing the constraint.

[2] This is similar to the experts model in online learning, hence the name.

Note that $\{\max_{j\in[m]} x_i(j,s) : i \in [n]\}$ is the *minimal* solution that is supported by the suggestions of expert s; hence, we define the cost of the solution proposed by expert s to be the cost of this solution.

The Multiple Predictions Setting. In the multiple predictions setting, we view the set of k suggestions in each step as a bag of k predictions (without indexing them specifically to individual predictors or experts) and the goal is to obtain a solution that can be benchmarked against the best of these suggestions in each step. Formally, our benchmark is the minimum-cost solution that is supported by at least one suggestion in each online step:

$$\text{DYNAMIC} = \min_{\hat{\mathbf{x}}\in\hat{X}} \sum_{i=1}^{n} c_i \cdot \hat{x}_i, \text{ where}$$

$$\hat{X} = \{\hat{\mathbf{x}} : \forall i \in [n], \forall j \in [m], \exists s \in [k], \hat{x}_i \geqslant x_i(j,s)\}.$$

Note that every solution that is supported in the experts setting is also supported in the multiple predictions setting. This implies that STATIC $\geqslant$ DYNAMIC, and therefore, the competitive ratios that we obtain in the multiple predictions setting also hold in the experts setting. Conversely, the lower bounds on the competitive ratio that we obtain in the experts setting also hold in the multiple predictions setting.

4.2.2 Our Results

We obtain an algorithm for OCP with the following guarantee in the multiple predictions setting (and therefore also in the experts setting by the discussion above):

Theorem 24. *There is an algorithm for the online covering problem with k suggestions that has a competitive ratio of $O(\log k)$, even against the* DYNAMIC *benchmark.*

Note that this competitive ratio is independent of the size of the problem instance, and only depends on the number of suggestions. In contrast, in the classical online setting, the competitive ratio (necessarily) depends on the size of the problem instance.

Next, we show that the competitive ratio in Theorem 24 is tight by showing a matching lower bound. This lower bound holds even in the experts setting (hence, by the discussion above, it automatically extends to the multiple predictions setting):

Theorem 25. *The competitive ratio of any algorithm for the online covering problem with k suggestions is $\Omega(\log k)$, even against the* STATIC *benchmark.*

We noted earlier that it is desirable for online algorithms to have *robustness* guarantees, i.e., that the algorithm does not fare much worse than the best online algorithm (without predictions) even if the predictions are completely inaccurate. Our next result is the robust version of Theorem 24:

Theorem 26. *Suppose a class of online covering problems have an online algorithm (without predictions) whose competitive ratio is α. Then, there is an algorithm for this class of online covering problems with k suggestions that produces an online solution whose cost is at most $O(\min\{\log k \cdot \text{DYNAMIC}, \alpha \cdot \text{OPT}\})$.*

We will prove Theorem 24 in the next section. The proofs of Theorem 25 and Theorem 26 are given in Section 4.3.2 and Section 4.3.1 respectively. Subsequently, we apply the algorithmic framework developed in Theorem 24 to obtain tight competitive ratios for specific instantiations of OCP, namely the set cover problem (Section 4.4) and the caching problem (Section 4.5). Finally, we extend our OCP result to include box-type constraints (Section 4.6) and apply it to the online facility location problem (Section 4.7).

4.3 Online Covering Algorithm

Recall that in the online covering problem, the new constraint that arrives in the j-th online step is $\sum_{i=1}^{n} a_{ij} x_i \geqslant 1$ and the algorithm receives k suggestions where the s-th suggestion is denoted $x_i(j,s)$. If the current solution of the algorithm given by the variables x_i is feasible, i.e., $\sum_{i=1}^{n} a_{ij} x_i \geqslant 1$, then the algorithm does not need to do anything. Otherwise,

the algorithm needs to increase these variables until they satisfy the constraint. Next, we describe the rules governing the increase of variables.

Intuitively, the rate of the increase of a variable x_i should depend on three things. First, it should depend on the cost of this variable in the objective, namely the value of c_i; the higher the cost, the slower we should increase this variable. Second, it should depend on the contribution of variable x_i in satisfying the new constraint, namely the value of a_{ij}; the higher this coefficient, the faster should we increase the variable. Finally, the third factor is how *strongly* x_i has been suggested. To encode this mathematically, we first make the assumption that every suggestion is *tight*, i.e.,

$$\sum_{i=1}^{n} a_{ij} x_i(j,s) = 1 \text{ for every suggestion } s \in [k]. \tag{4.1}$$

This assumption is without loss of generality because, if not, we can decrease the variables $x_i(j,s)$ in an arbitrary manner until the constraint becomes tight. (Note that this change can only decrease the cost of the benchmark solutions DYNAMIC and STATIC; hence, any competitive ratio bounds obtained after this transformation also hold for the original set of suggestions.)

Having made all the suggestions tight, we now encode how strongly a variable has been suggested by using its *average* suggested value $\frac{1}{k} \cdot \sum_{s=1}^{k} x_i(j,s)$. Our algorithm (see Algorithm 11) increases all variables x_i satisfying $x_i < \frac{1}{2}$ simultaneously at rates governed by these parameters; precisely, we use

$$\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}\left(x_i + \delta \cdot x_{ij}\right), \text{ where } \delta = \frac{1}{k}, x_{ij} = \sum_{s=1}^{k} x_i(j,s).$$

The algorithm continues to increase the variables until $\sum_{i=1}^{n} a_{ij} x_i \geqslant \frac{1}{2}$; along the way, any variable x_i that reaches $\frac{1}{2}$ is dropped from the set of increasing variables. To satisfy the j-th constraint, we note that the variables $2x_i$ are feasible for the constraint. (Note that since

all variables $x_i \leqslant \frac{1}{2}$ before the scaling, every variable can be doubled without violating $x_i \leqslant 1$.) Since this last step of multiplying every variable by 2 only increases the cost of the algorithm by a factor of 2, we ignore this last scaling step in the rest of the analysis.

Algorithm 11 Online Covering Algorithm

Offline: All variables x_i are initialized to 0.
Online: On arrival of the j-th constraint:
  while $\sum_{i=1}^{n} a_{ij} x_i < \frac{1}{2}$,
    for $i \in [n]$
      if $x_i < \frac{1}{2}$, increase x_i by $\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}\left(x_i + \delta \cdot x_{ij}\right)$,
        where $\delta = \frac{1}{k}$ and $x_{ij} = \sum_{s=1}^{k} x_i(j,s)$.

Before analyzing the algorithm, we note that although we described it using a continuous process driven by a differential equation, the algorithm can be easily discretized and made to run in polynomial time where in each discrete step, some variable x_i reaches $\frac{1}{2}$ (and therefore, x_i cannot increase any further) or $\sum_{i=1}^{n} a_{ij} x_i$ reaches $\frac{1}{2}$ (and therefore, the algorithm ends for the current online step). In this section, we will analyze the continuous algorithm rather than the equivalent discrete algorithm for notational simplicity.

Next, we show that the algorithm is valid, i.e., that there is always a variable x_i that can be increased inside the **while** loop. If not, then we have $\sum_{i=1}^{n} a_{ij} x_i < \frac{1}{2}$ but $x_i \geqslant \frac{1}{2}$ for all variables x_i, $i \in [n]$. This implies that $\sum_{i=1}^{n} a_{ij} < 1$, which is a contradiction because the constraint $\sum_{i=1}^{n} a_{ij} x_i \geqslant 1$ is then unsatisfiable by any setting of variables $x_i \leqslant 1$. (In particular, this would mean that there cannot be any feasible suggestion for this constraint.)

Now, we are ready to bound the competitive ratio of Algorithm 11 with respect to the DYNAMIC benchmark. First, we bound the rate of increase of algorithm's cost:

Lemma 28. *The rate of increase of cost in Algorithm 11 is at most* $\frac{3}{2}$.

Proof. The rate of increase of cost is given by:

$$\begin{aligned}\sum_{i=1}^{n} c_i \cdot \frac{dx_i}{dt} &= \sum_{i=1}^{n} a_{ij}\left(x_i + \delta \cdot x_{ij}\right) \\ &= \sum_{i=1}^{n} a_{ij}x_i + \frac{1}{k} \cdot \sum_{i=1}^{n}\sum_{s=1}^{k} a_{ij}x_i(j,s) \\ &< \frac{1}{2} + \frac{1}{k} \cdot \sum_{s=1}^{k}\left(\sum_{i=1}^{n} a_{ij}x_i(j,s)\right) = \frac{3}{2},\end{aligned}$$

where we used $\sum_{i=1}^{n} a_{ij}x_i < \frac{1}{2}$ from the condition on the **while** loop, and $\sum_{i=1}^{n} a_{ij}x_i(j,s) = 1$ for all $s \in [k]$ from Equation (4.1). □

We now define a carefully crafted non-negative potential function ϕ. We will show that the potential decreases at constant rate when Algorithm 11 increases the variables x_i (Lemma 31). By Lemma 28, this implies that the potential can pay for the cost of Algorithm 11 up to a constant. We will also show that the potential ϕ is at most $O(\log k)$ times the DYNAMIC benchmark (Lemma 30). Combined, these yield Theorem 24.

Let x_i^{DYN} denote the value of variable x_i in the DYNAMIC benchmark. The potential function for a variable x_i is then defined as follows:

$$\phi_i = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}}, \text{ where } \delta = \frac{1}{k}.$$

and the overall potential is:

$$\phi = \sum_{i : x_i^{\text{DYN}} \geqslant x_i} \phi_i.$$

The intuition behind only including those variables that have $x_i^{\text{DYN}} \geqslant x_i$ in the potential function is that the potential stores the excess cost paid by the DYNAMIC benchmark for these variables so that it can be used later to pay for increase in the algorithm's variables.

First, we verify that the potential function is always non-negative.

Lemma 29. *For any values x_i, x_i^{DYN} of the variables, the potential function ϕ is non-negative.*

Proof. Note that ϕ only includes variables x_i such that $x_i^{\text{DYN}} \geqslant x_i$. For such variables,

$$\phi_i = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{1+\delta}{\frac{x_i}{x_i^{\text{DYN}}} + \delta} \geqslant 0. \qquad \square$$

Next, we bound the potential as a function of the variables x_i^{DYN} in the DYNAMIC benchmark:

Lemma 30. *The potential ϕ_i for variable x_i is at most $c_i x_i^{\text{DYN}} \cdot \ln\left(1 + \frac{1}{\delta}\right) = c_i x_i^{\text{DYN}} \cdot O(\log k)$. As a consequence, the overall potential $\phi \leqslant O(\log k) \cdot \sum_{i=1}^{n} c_i x_i^{\text{DYN}}$.*

Proof. We have

$$\begin{aligned}
\phi_i &= c_i x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} \\
&\leqslant c_i x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{\delta x_i^{\text{DYN}}} \\
&= c_i x_i^{\text{DYN}} \cdot \ln\left(1 + \frac{1}{\delta}\right) \\
&= c_i x_i^{\text{DYN}} \cdot O(\log k). \qquad \square
\end{aligned}$$

Finally, we bound the rate of decrease of potential ϕ with increase in the variables x_i in Algorithm 11. Our goal is to show that up to constant factors, the decrease in potential ϕ can pay for the increase in cost of the solution of Algorithm 11.

Lemma 31. *The rate of decrease of the potential ϕ with increase in the variables x_i in Algorithm 11 is at least $\frac{1}{2}$.*

Proof. Recall that $\phi_i = c_i \cdot x_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}}$. Therefore,

$$\frac{d\phi_i}{dx_i} = -c_i \cdot \frac{x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}}.$$

Recall that in Algorithm 11, the rate of increase of variables x_i is given by $\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}\left(x_i + \delta \cdot x_{ij}\right)$, where $\delta = \frac{1}{k}$ and $x_{ij} = \sum_{s=1}^{k} x_i(j,s)$. Thus, we have:

$$\frac{d\phi_i}{dt} = \frac{d\phi_i}{dx_i} \cdot \frac{dx_i}{dt} = -c_i \cdot \frac{x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} \cdot \frac{a_{ij}}{c_i}(x_i + \delta \cdot x_{ij})$$

$$= -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta x_{ij}}{x_i + \delta x_i^{\text{DYN}}}.$$

Now, we have two cases:

- If $x_{ij} \geqslant x_i^{\text{DYN}}$, then

$$\frac{d\phi_i}{dt} = -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta x_{ij}}{x_i + \delta x_i^{\text{DYN}}}$$

$$\leqslant -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} = -a_{ij} x_i^{\text{DYN}}. \tag{4.2}$$

- If $x_{ij} < x_i^{\text{DYN}}$, then

$$\frac{d\phi_i}{dt} = -a_{ij} \cdot x_i^{\text{DYN}} \cdot \frac{x_i + \delta \cdot x_{ij}}{x_i + \delta x_i^{\text{DYN}}}$$

$$= -a_{ij} \cdot x_{ij} \cdot \frac{\frac{x_i^{\text{DYN}}}{x_{ij}} \cdot x_i + \delta \cdot x_i^{\text{DYN}}}{x_i + \delta x_i^{\text{DYN}}} < -a_{ij} x_{ij}. \tag{4.3}$$

We know that at least one of the suggestions in the j-th step is supported by the DYNAMIC benchmark. Let $s(j) \in [k]$ be such a supported suggestion. Then,

$$x_{ij} = \sum_{s=1}^{k} x_i(j,s) \geqslant x_i(j,s(j)), \text{ and}$$

$$x_i^{\text{DYN}} \geqslant x_i(j,s(j)) \text{ since } s(j) \text{ is supported by DYNAMIC.}$$

Therefore, in both cases (Equation (4.2) and Equation (4.3)) above, we get

$$\frac{d\phi_i}{dt} \leqslant -a_{ij} x_i(j,s(j)).$$

Let us denote $I_j = \{i \mid x_i(j, s(j)) \geqslant x_i\}$. Then, the total decrease in potential is given by:

$$\frac{d\phi}{dt} = \sum_{i : x_i^{\text{DYN}} \geqslant x_i} \frac{d\phi_i}{dt} \leqslant -\sum_{i \in I_j} a_{ij} x_i(j, s(j)). \tag{4.4}$$

By feasibility of the $s(j)$-th suggestion for the j-th constraint, we have $\sum_{i=1}^{n} a_{ij} x_i(j, s(j)) \geqslant 1$. Therefore,

$$\sum_{i \in I_j} a_{ij} x_i(j, s(j)) + \sum_{i \notin I_j} a_{ij} x_i(j, s(j)) \geqslant 1$$

$$\text{i.e., } \sum_{i \in I_j} a_{ij} x_i(j, s(j)) + \sum_{i \notin I_j} a_{ij} x_i > 1,$$

since $x_i > x_i(j, s(j))$ for $i \notin I_j$. Thus,

$$\sum_{i \in I_j} a_{ij} x_i(j, s(j)) + \sum_{i} a_{ij} x_i > 1$$

$$\text{i.e., } \sum_{i \in I_j} a_{ij} x_i(j, s(j)) > \frac{1}{2},$$

since $\sum_{i=1}^{n} a_{ij} x_i < \frac{1}{2}$ in Algorithm 11. The lemma follows by Equation (4.4). □

Theorem 24 now follows from the above lemmas using standard arguments as follows:

Proof of Theorem 24. Initially, let $x_i = 0$ for all $i \in [n]$ but let x_i^{DYN} be their final value. Then, by Lemma 30, the potential ϕ is at most $O(\log k)$ times the cost of DYNAMIC. Now, as Algorithm 11 increases the values of the variables x_i, it incurs cost at rate at most $\frac{3}{2}$ (by Lemma 28) and the potential ϕ decreases at rate at least $\frac{1}{2}$ (by Lemma 31). Since ϕ is always non-negative (by Lemma 29), it follows that the total cost of the algorithm is at most 3 times the potential ϕ at the beginning, i.e., at most $O(\log k)$ times the DYNAMIC benchmark. This completes the proof of Theorem 24. □

4.3.1 Robust Algorithm for the Online Covering Problem

Now, we prove the robust version of Theorem 24, namely Theorem 26.

Proof of Theorem 26. We run a meta algorithm with two sets of suggestions corresponding to two solutions. The first solution is obtained by using Algorithm 11 with k suggestions. By Theorem 24 this solution has cost at most $O(\log k) \cdot \text{DYNAMIC}$. The second solution is produced by the online algorithm that achieves a competitive ratio of α in the statement of Theorem 26. Using Algorithm 11 again for the meta algorithm, Theorem 26 now follows by invoking Theorem 24. □

As one particular application of Theorem 26, we note that for general OCP, the best competitive ratio is due to the following result of Gupta and Nagarajan Gupta and Nagarajan (2014) (see also Buchbinder and Naor Buchbinder and Naor (2009b)):

Theorem 27 (Gupta and Nagarajan Gupta and Nagarajan (2014))**.** *There is an algorithm for the online covering problem that has a competitive ratio of $O(\log d)$, where d is the maximum number of variables with non-zero coefficients in any constraint.*

This automatically implies the following corollary of Theorem 26:

Theorem 28. *There is an algorithm for the fractional online covering problem that produces an online solution whose cost is at most $O(\min\{\log k \cdot \text{DYNAMIC}, \ln d \cdot \text{OPT}\})$ in the multiple predictions setting with k predictions, where d is the maximum number of non-zero coefficients in any constraints of the online covering problem instance.*

For specific problems that can be modeled as OCP, it might be possible to obtain a better competitive ratio than $O(\log d)$ by using the structure of those instances. In that case, the competitive ratio in the multiple predictions setting also improves accordingly by Theorem 26.

4.3.2 *Lower Bound for the Online Covering Problem*

Here we show that the competitive ratio obtained in Theorem 24 is tight, i.e., we prove Theorem 25. We will restrict ourselves to instances of OCP where $a_{ij} \in \{0,1\}$ and $c_i = 1$ for all i, j; this is called the *online (fractional) set cover* problem. (We will discuss the set cover problem in more detail in the next section.) Moreover, our lower bound will hold even in the experts model, i.e., against the STATIC benchmark. Since the DYNAMIC benchmark is always at most the STATIC benchmark, it follows that the lower bound also holds for the DYNAMIC benchmark.

Proof of Theorem 25. We index the k experts $\{1, 2 \dots k\}$ using a uniform random permutation. We will construct an instance of OCP, where the cost of the optimal solution is T. The instance has k rounds, where in each round there are T constraints. We index the jth constraint of the ith round as (i, j) for $i \in [k], j \in [T]$. There are kT variables that are also indexed as (i, j) for $i \in [k], j \in [T]$. Constraint (i, j) is satisfied by each of the variables (i', j) for all $i' \geqslant i$ (i.e., $a_{(i,j),(i',j)} = 1$). When constraint (i, j) is presented (in round i), expert i' for every $i' \geqslant i$ sets variable (i', j) to 1 to satisfy it. (The suggestions of experts $i'' < i$ are immaterial in this round, and they can set any arbitrary variable satisfying constraint (i, j) to 1.)

The optimal solution is to follow expert k, i.e., the variables (k, j) for all $j \in [T]$ should be set to 1; this has cost T. After round i, the cumulative expected cost of any deterministic algorithm across the variables $(i,1), (i,2), \dots, (i,T)$ is at least $\frac{T}{i}$. Across all $i \in [k]$, this adds up to a total expected cost $T \cdot \left(1 + \frac{1}{2} + \dots + \frac{1}{k}\right) = \Omega(T \log k)$. The theorem then follows by Yao's minimax principle Yao (1977). □

4.4 Online Set Cover

In the (weighted) set cover problem, we are given a collection of subsets $\mathscr{S}$ of a ground set U, where set $S \in \mathscr{S}$ has weight w_S. The goal is to select a collection of sets $\mathscr{T} \subseteq \mathscr{S}$ of

minimum cost $\sum_{T \in \mathscr{T}} w_T$ that cover all the elements in U, i.e., that satisfies $\cup_{T \in \mathscr{T}} T = U$. In the *online* version of this problem (see Alon et al. (2009)), the set of elements in U is not known in advance. In each online step, a new element $u \in U$ is revealed, and the sets in $\mathscr{S}$ that contain u are identified. If u is not covered by the sets in the current solution $\mathscr{T}$, then the algorithm must *augment* $\mathscr{T}$ by adding some set containing u to it.

In the fractional version of the set cover problem, sets can be selected to fractions in $[0,1]$, i.e., a solution is given by variables $x_S \in [0,1]$ for all $S \in \mathscr{S}$. The constraint is that the total fraction of all sets containing each element u must be at least 1, i.e., $\sum_{S:u \in S} x_S \geqslant 1$ for every element $u \in U$. The cost of the solution is given by $\sum_{S \in \mathscr{S}} w_S x_S$. Clearly, this is a special case of the online covering problem in the previous section where each variable x_i represents a set and each constraint $\sum_{i=1}^{n} a_{ij} x_i \geqslant 1$ is for an element, where $a_{ij} = 1$ if an only if element j is in set i, else $a_{ij} = 0$.

We define the *frequency* of an element to be the number of sets containing it, and denote the maximum frequency of any element by d. Note that this coincides with the maximum number of non-zero coefficients in any constraint. The following theorem is an immediate corollary of Theorem 28:

Theorem 29. *There is an algorithm for the fractional online set cover problem that produces an online solution whose cost is at most* $O(\min\{\ln k \cdot \text{DYNAMIC}, \ln d \cdot \text{OPT}\})$ *in the multiple predictions setting with k predictions.*

It is interesting to note that when the suggestions are good in the sense that $\text{DYNAMIC} = O(\text{OPT})$, the competitive ratio of $O(\log k)$ in the above theorem is independent of the size of the problem instance. In contrast, for the classical fractional online set cover problem, there is a well-known lower bound of $\Omega(\log d)$ on the competitive ratio. We also note that the competitive ratio in Theorem 29 is tight since as we noted earlier, the lower bound instance constructed in Theorem 25 is actually an instance of the set cover problem.

4.4.1 Online Rounding and the Integral Set Cover Problem

Next, we consider the integral set cover problem, i.e., where the variables x_i are required to be integral, i.e., in $\{0,1\}$ rather than in $[0,1]$. The following a well-known result on rounding fractional set cover solutions online:

Theorem 30 (Alon *et al.* Alon et al. (2009))**.** *Given any feasible solution to the fractional online set cover problem, there is an online algorithm for finding a feasible solution to the integral online set cover problem whose cost is at most $O(\log m)$ times that of the fractional solution, where m is the number of elements.*

By applying this theorem on the fractional solution produced by Theorem 29, we get a competitive ratio of $O(\log m \log k)$ for the integral online set cover problem with k predictions against the DYNAMIC benchmark:

Theorem 31. *There is an algorithm for the integral online set cover problem that produces an online solution whose cost is at most $O(\min\{\ln m \ln k \cdot \text{DYNAMIC}, \ln m \ln d \cdot \text{OPT}\})$ in the multiple predictions setting with k predictions.*

It is well-known that the competitive ratio of the integral online set cover problem (without predictions) is at least $\tilde{\Omega}(\log m \log d)$ Alon et al. (2009)[3]. In the degenerate case of $k = d$, any instance of online set cover can be generated in the k predictions settings where the benchmark solution DYNAMIC will be the optimal solution, since all the sets containing an element can be provided as suggestions in each online step. As a consequence, the competitive ratio in Theorem 31 is tight (up to lower order terms).

4.5 (Weighted) Caching

The caching problem is among the most well-studied online problems (see, e.g., Sleator and Tarjan (1985); Fiat et al. (1991); McGeoch and Sleator (1991); Achlioptas et al. (2000);

[3] The notation $\tilde{\Omega}$ (and $\tilde{O}$) hide lower order terms.

Young (1991); Blum et al. (1999), and the textbook Borodin and El-Yaniv (1998)). In this problem, there is a set of n pages and a cache that can hold any h pages at a time.[4] In every online step j, a page p_j is requested; if this page is not in the cache, then it has to be brought into the cache (called *fetching*) by evicting an existing page from the cache. In the weighted version (see, e.g., Chrobak et al. (1991); Young (1994); Bansal et al. (2007)), the cost of fetching a page p into the cache is given by its non-negative weight w_p. (In the unweighted version, $w_p = 1$ for all pages.) The goal of a caching algorithm is to minimize the total cost of fetching pages while serving all requests.

The (weighted) caching problem can be formulated as online covering problem by defining variables $x_p(r) \in \{0,1\}$ to indicate whether page p is evicted between its r-th and $(r+1)$-st requests. Let $r(p,j)$ denote the number of times page p is requested until (and including) the j-th request. For any online step j, let $B(j) = \{p : r(p,j) \geqslant 1\}$ denote the set of pages that have been requested until (and including) the j-th request. The covering program formulation is as follows:

$$\begin{array}{ll} \min & \sum_p \sum_r w_p \cdot x_p(r) \text{ subject to} \\ & \sum_{p \in B(j), p \neq p_j} x_p(r(p,j)) \geqslant |B(j)| - h, \; \forall j \geqslant 1 \\ & x_p(r) \in \{0,1\}, \qquad\qquad \forall p, \forall r \geqslant 1 \end{array}$$

In the fractional version of the problem, we replace the constraints $x_p(r) \in \{0,1\}$ with the constraints $x_p(r) \in [0,1]$. Clearly, this fits the definition of the online covering problem.[5] Moreover, for the fractional weighted caching problem, Bansal, Buchbinder, and Naor gave an online algorithm with a competitive ratio of $O(\log h)$:

Theorem 32 (Bansal et al. (2007))**.** *There is an online algorithm for the fractional weighted caching problem with a competitive ratio of* $O(\log h)$.

[4] The usual notation for cache size is k, but we have changed it to h since we are using k to denote the number of suggestions.

[5] Strictly speaking, we need to scale the first set of coefficients by $|B(j)| - h$, but as we mentioned earlier, this is equivalent since scaling the coefficients has no bearing on the competitive ratio of our algorithm.

Note that the competitive ratio of $O(\log h)$ is better than that given by Theorem 27 since the cache size h is typically much smaller than the total number of pages. Now, we apply Theorem 26 to get the following result for fractional weighted caching with k predictions:

Theorem 33. *There is an algorithm for the fractional weighted caching problem that produces an online solution whose cost is at most* $O(\min\{\ln k \cdot \text{DYNAMIC}, \ln h \cdot \text{OPT}\})$ *in the multiple predictions setting with k predictions.*

4.5.1 Integral (Weighted) Caching

Next, we consider the integral weighted caching problem, i.e., where the variables $x_p(r)$ have to be in $\{0,1\}$ rather than $[0,1]$. We use the following known result about online rounding of fractional weighted caching solutions:

Theorem 34 (Bansal, Buchbinder, and Naor Bansal et al. (2007))**.** *Given any feasible online solution to the fractional weighted caching problem, there is an online algorithm for finding a feasible online solution to the integral weighted caching problem whose cost is at most* $O(1)$ *times that of the fractional solution.*

By applying this theorem on the fractional solution produced by Theorem 33, we get a competitive ratio of $O(\log k)$ for the integral weighted caching problem with k predictions against the DYNAMIC benchmark:

Theorem 35. *There is an online algorithm for the integral weighted caching problem that produces an online solution whose cost is at most* $O(\min\{\ln k \cdot \text{DYNAMIC}, \ln h \cdot \text{OPT}\})$ *in the multiple predictions setting with k predictions.*

4.6 Online Covering with Box Constraints

In this section, we generalize the online covering framework in Section 4.2 by allowing additional *box* constraints of the form $x_{ij} \leqslant y_i$. The new linear program is given by:

$$\text{Minimize } \sum_{i=1}^{n} c_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} x_{ij} \text{ subject to}$$

$$x_{ij} \leqslant y_i \quad \forall i, j \tag{4.5}$$

$$\sum_{i=1}^{n} a_{ij} x_{ij} = 1 \quad \forall j \tag{4.6}$$

$$y_i \in [0,1] \quad \forall i \tag{4.7}$$

The box constraints $x_{ij} \leqslant y_i$ do not have coefficients and hence are known offline. As in OCP, the cost coefficients c_i are known offline. In each online step, a new covering constraint $\sum_{i=1}^{n} a_{ij} x_{ij} \geqslant 1$ is revealed to the algorithm, and the corresponding cost coefficient d_{ij} is also revealed.

We first note that this is a generalization of the online covering problem. To see this, set $d_{ij} = 0$. Then, we get:

$$\min_{y_i \in [0,1]: i \in [n]} \left\{ \sum_{i=1}^{n} c_i y_i : \sum_{i=1}^{n} a_{ij} y_i \geqslant 1 \ \forall j \in [m] \right\},$$

which is precisely the online covering problem.

The more general version captures problems like facility location (shown in the next section) that are not directly modeled by OCP. We denote the s-th suggestion for the j-th constraint by variables $y_i(j,s)$ and $x_{ij}(s)$; they satisfy $\sum_{i=1}^{n} a_{ij} x_{ij}(s) \geqslant 1$, i.e., all suggestions are feasible.

4.6.1 Online Algorithm

Our algorithm for OCP with box constraints is given in Algorithm 12. As earlier, for all i, we simultaneously raise x_{ij} (and possibly y_i as well) at the rate specified by the algorithm.

As in the case of OCP, we raise these variables only till the constraint is satisfied up to a factor of $\frac{1}{2}$, and any individual variable does not cross $\frac{1}{2}$. This allows us to double all variable and satisfy the constraints at an additional factor of 2 in the cost. Moreover, the same argument as in Algorithm 11 implies that this algorithm is also feasible, i.e., there is at least one variable that can be increased in the **while** loop.

Algorithm 12 Algorithm for Online FACILITY LOCATION

On arrival of a new constraint $\sum_{i=1}^{n} a_{ij}x_{ij} = 1$:
  Initialize $x_{ij} = 0, \quad \forall j$.
  Set $\Gamma_{ij} := \sum_{s=1}^{k} x_{ij}(s), \quad \forall j$ and $\delta = \frac{1}{k}$.
    while $\sum_j x_{ij} < \frac{1}{2}$
      for all i such that $x_{ij} < \frac{1}{2}$:
        if $x_{ij} < y_i$
          Increase x_{ij} at the rate $\frac{\partial x_{ij}}{\partial t} = \left(\frac{a_{ij}}{d_{ij}}\right) \cdot \left(x_{ij} + \delta \cdot \Gamma_{ij}\right)$
        else
          Increase both variables x_{ij}, y_i at the same rate
          $\frac{\partial y_i}{\partial t} = \frac{\partial x_{ij}}{\partial t} = \left(\frac{a_{ij}}{d_{ij}+c_i}\right) \cdot \left(x_{ij} + \delta \cdot \Gamma_{ij}\right)$.

4.6.2 Analysis

In this section, we analyze Algorithm 12. We first bound the rate of increase of the cost of the algorithm.

Lemma 32. *The rate of increase of the cost for Algorithm 12 is at most* $\frac{3}{2}$.

Proof. Consider Algorithm 12 when the constraint $\sum_{i=1}^{n} a_{ij}x_{ij} = 1$ arrives. For any i, we claim:

$$d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} + c_i \cdot \frac{\partial y_i}{\partial t} = a_{ij}\left(x_{ij} + \delta \cdot \Gamma_{ij}\right). \tag{4.8}$$

To show this, there are two cases to consider:

- $x_{ij} < y_i$: In this case, $\frac{\partial y_i}{\partial t} = 0$, and $d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} = a_{ij}\left(x_{ij} + \delta \cdot \Gamma_{ij}\right)$ and so (4.8) holds.

- $x_{ij} = y_i$: In this case, $\frac{\partial y_i}{\partial t} = \frac{\partial x_{ij}}{\partial t} = \frac{a_{ij}}{d_{ij}+c_i} \cdot \left(x_{ij} + \frac{\Gamma_{ij}}{k}\right)$, i.e., $c_i \cdot \frac{\partial y_i}{\partial t} + d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} = a_{ij}\left(x_{ij} + \delta \cdot \Gamma_{ij}\right)$, and so (4.8) holds.

Therefore, the rate of change of the objective function is given by:

$$\sum_i \left(d_{ij} \cdot \frac{\partial x_{ij}}{\partial t} + c_i \cdot \frac{\partial y_i}{\partial t} \right) \overset{(4.8)}{=} \sum_i a_{ij}\left(x_{ij} + \delta \cdot \Gamma_{ij}\right) = \sum_i a_{ij} x_{ij} + \frac{\sum_s \sum_i a_{ij} x_{ij}(s)}{k} \leqslant \frac{1}{2} + 1 = \frac{3}{2}.$$

□

We now describe the potential function, which has a similar structure as that in the online covering framework. Let $x_{ij}^{\text{DYN}}, y_i^{\text{DYN}}$ denote the values of the variables x_{ij}, y_i respectively in the benchmark solution DYNAMIC. The potential function for a variable x_{ij} is then defined as follows:

$$\phi_{ij} = d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}, \text{ where } \delta = \frac{1}{k},$$

and the potential for the variable y_i is given by

$$\psi_i = c_i \cdot y_i^{\text{DYN}} \cdot \ln \frac{(1+\delta)y_i^{\text{DYN}}}{y_i + \delta y_i^{\text{DYN}}}.$$

As before, the overall potential is

$$\phi = \sum_{i,j : x_{ij}^{\text{DYN}} \geqslant x_{ij}} \phi_{ij} + \sum_{i : y_i^{\text{DYN}} \geqslant y_i} \psi_i.$$

The rest of the proof proceeds along the same lines as that for the online covering problem. But we give the details for the sake of completeness.

The next lemma is the analogue of Lemma 29, and shows that the potential ϕ is always non-negative.

Lemma 33. *For any values of the variables $x_{ij}, y_i, x_{ij}^{\text{DYN}}, y_i^{\text{DYN}}$, the potential function ϕ is non-negative.*

Proof. We show that each of the quantities ϕ_{ij}, ψ_i in the expression for ϕ is non-negative. Consider a pair i, j for which $x_{ij}^{\text{DYN}} \geqslant x_{ij}$. Then

$$\phi_{ij} = d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}} = d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln \frac{1+\delta}{\frac{x_{ij}}{x_{ij}^{\text{DYN}}} + \delta} \geqslant 0.$$

Similarly, $\psi_i \geqslant 0$ if $y_i^{\text{DYN}} \geqslant y_i$. This shows that $\phi \geqslant 0$. □

Now we bound the potential against the benchmark solution DYNAMIC. The proof of this lemma is very similar to that of Lemma 30.

Lemma 34. *The following bounds hold:* $\phi_{ij} \leqslant d_{ij} \cdot x_{ij}^{\text{DYN}} \cdot \ln(1 + \frac{1}{\delta}) = d_{ij} x_{ij}^{\text{DYN}} \cdot O(\log k)$ *and* $\psi_i \leqslant c_i \cdot y_i^{\text{DYN}} \cdot \ln(1 + \frac{1}{\delta})) = c_i y_i^{\text{DYN}} \cdot O(\log k)$. *As a consequence, the overall potential* $\phi \leqslant O(\log k) \cdot \left(\sum_i \sum_j d_{ij} x_{ij}^{\text{DYN}} + \sum_j c_i y_i^{\text{DYN}} \right)$.

Proof. We have

$$\phi_{ij} = d_{ij} x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{x_i + \delta x_{ij}^{\text{DYN}}} \leqslant d_{ij} x_{ij}^{\text{DYN}} \cdot \ln \frac{(1+\delta)x_{ij}^{\text{DYN}}}{\delta x_{ij}^{\text{DYN}}} = d_{ij} x_{ij}^{\text{DYN}} \cdot \ln \left(1 + \frac{1}{\delta}\right) = d_{ij} x_{ij}^{\text{DYN}} \cdot O(\log k).$$

The bound for ψ_i also follows similarly. □

Finally, we bound the rate of decrease of potential ϕ with increase in the variables in Algorithm 12.

Lemma 35. *The rate of decrease of the potential* ϕ *in Algorithm 12 is at least* $\frac{1}{2}$.

Proof. It is easy to check that

$$\frac{\partial \phi_{ij}}{\partial x_{ij}} = -\frac{d_{ij} x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}, \quad \frac{\partial \psi_i}{\partial y_i} = -\frac{c_i y_i^{\text{DYN}}}{y_i + \delta y_i^{\text{DYN}}}. \tag{4.9}$$

Consider the step when the j-th constraint arrives. We claim that for any index $i \in [n]$,

$$\frac{\partial \left(\phi_{ij} + \psi_i\right)}{\partial t} \leqslant -a_{ij} x_{ij}^{\text{DYN}} \cdot \frac{x_{ij} + \delta \Gamma_{ij}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}. \tag{4.10}$$

To prove this, we consider two cases:

- $x_{ij} < y_i$: In this case, $\frac{\partial x_{ij}}{\partial t} = \frac{a_{ij}}{d_{ij}} \cdot \left(x_{ij} + \delta\Gamma_{ij}\right)$, $\frac{\partial y_i}{\partial t} = 0$. Combining this with (4.9), we see that

$$\frac{\partial\left(\phi_{ij} + \psi_i\right)}{\partial t} = \frac{\partial \phi_{ij}}{\partial t} = \frac{\partial \phi_{ij}}{\partial x_{ij}} \cdot \frac{\partial x_{ij}}{\partial t} = -a_{ij} x_{ij}^{\text{DYN}} \cdot \frac{x_{ij} + \delta\Gamma_{ij}}{x_{ij} + \delta x_{ij}^{\text{DYN}}}.$$

- $x_{ij} = y_i$: In this case, $\frac{\partial x_{ij}}{\partial t} = \frac{\partial y_i}{\partial t} = \frac{a_{ij}}{d_{ij}+c_i} \cdot \left(x_{ij} + \frac{\Gamma_{ij}}{k}\right)$. Using (4.9) again and the fact that $y_i = x_{ij}$, we see that

$$\frac{\partial\left(\phi_{ij} + \psi_i\right)}{\partial t} = \frac{\partial \phi_{ij}}{\partial x_{ij}} \cdot \frac{\partial x_{ij}}{\partial t} + \frac{\partial \psi_i}{\partial y_i} \cdot \frac{\partial y_i}{\partial t} = -\frac{a_{ij}}{c_i + d_{ij}} \left(\frac{d_{ij} x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}} + \frac{c_i y_i^{\text{DYN}}}{x_{ij} + \delta y_i^{\text{DYN}}}\right) \cdot \left(x_{ij} + \delta\Gamma_{ij}\right).$$

  Since $y_i^{\text{DYN}} \geqslant x_{ij}^{\text{DYN}}$, $\frac{y_i^{\text{DYN}}}{x_{ij}+\delta y_i^{\text{DYN}}} \geqslant \frac{x_{ij}^{\text{DYN}}}{x_{ij}+\delta x_{ij}^{\text{DYN}}}$. Therefore, the RHS above is at most $-a_{ij} x_{ij}^{\text{DYN}} \cdot \frac{x_{ij}+\delta\Gamma_{ij}}{x_{ij}+\delta x_{ij}^{\text{DYN}}}$.

Thus, we have shown that inequality (4.10) always holds. Now, we have two cases:

- $\Gamma_{ij} \geqslant x_{ij}^{\text{DYN}}$: Inequality (4.10) implies that

$$\frac{\partial(\phi_{ij} + \psi_i)}{\partial t} \leqslant -a_{ij} x_{ij}^{\text{DYN}}.$$

- $\Gamma_{ij} < x_{ij}^{\text{DYN}}$: Using (4.10) again, we see that

$$\frac{d(\phi_{ij} + \psi_i)}{dt} \leqslant -a_{ij} x_{ij}^{\text{DYN}} \cdot \frac{x_{ij} + \delta \cdot \Gamma_{ij}}{x_{ij} + \delta x_{ij}^{\text{DYN}}} = -a_{ij}\Gamma_{ij} \cdot \frac{\frac{x_{ij}^{\text{DYN}}}{\Gamma_{ij}} \cdot x_{ij} + \delta \cdot x_{ij}^{\text{DYN}}}{x_{ij} + \delta x_{ij}^{\text{DYN}}} \leqslant -a_{ij}\Gamma_{ij}.$$

We know that at least one of the suggestions in the i-th step is supported by the DYNAMIC benchmark. Let $s(j) \in [k]$ be the index of such a supported suggestion. Then,

$$\Gamma_{ij} = \sum_{s=1}^{k} x_{ij}(s) \geqslant x_{ij}(s(j)), \text{ and}$$

$$x_{ij}^{\text{DYN}} \geqslant x_{ij}(s(j)) \text{ since } s(j) \text{ is supported by DYNAMIC.}$$

Therefore, in both cases above, we get

$$\frac{\partial(\phi_{ij} + \psi_i)}{\partial t} \leqslant -a_{ij} \cdot x_{ij}(s(j)).$$

Let I denote the index set $\{i \in [n] : x_{ij}^{\text{DYN}} \geqslant x_{ij}\}$ (here j is fixed). We claim that the total decrease in potential satisfies:

$$\frac{\partial \phi}{\partial t} \leqslant -\sum_{i \in I} a_{ij} \cdot x_{ij}(s(j)). \tag{4.11}$$

To see this, let I' denote the index set $\{i : y_i^{\text{DYN}} \geqslant y_i\}$. Then the term in ϕ corresponding to step j is $\phi_j := \sum_{i \in I} \phi_{ij} + \sum_{i \in I'} \psi_i$. First consider an index $i \in I$ for which $x_{ij} < y_i$. In this case, we know that $\frac{\partial \psi_i}{\partial t} = 0$, and so irrespective of whether i belongs to I' or not, the rate of change of the terms in ϕ_j corresponding to i is

$$\frac{\partial(\phi_{ij} + \psi_i)}{\partial t} \leqslant -a_{ij} \cdot x_{ij}(s(j)).$$

Now consider an index $i \in I$ for which $x_{ij} = y_i$. Since $y_i^{\text{DYN}} \geqslant x_{ij}^{\text{DYN}}$, it follows that $y_i^{\text{DYN}} \geqslant y_i$ and so $i \in I'$ as well. Therefore, the rate of change of the terms in ϕ_j corresponding to i is equal to

$$\frac{\partial(\phi_{ij} + \psi_i)}{\partial t} \leqslant -a_{ij} \cdot x_{ij}(s(j)).$$

Finally, consider an index $i \in I' \backslash I$. It is easy to verify that $\frac{\partial \psi_i}{\partial t} \leqslant 0$. Thus, inequality (4.11) follows.

The rest of the argument proceeds as in the proof of Lemma 31. By feasibility of the

$s(j)$-th suggestion in step j, we have:

$$\sum_j a_{ij} x_{ij}(s(j)) \geqslant 1$$

$$\text{i.e., } \sum_{i \in I} a_{ij} x_{ij}(s(j)) + \sum_{i \notin I} a_{ij} x_{ij}(s(j)) \geqslant 1$$

$$\text{i.e., } \sum_{i \in I} a_{ij} x_{ij}(s(j)) + \sum_{i \notin I} a_{ij} x_{ij} > 1 \qquad \left(\text{since } i \notin I, \text{ we have } x_{ij} > x_{ij}^{\text{DYN}} \geqslant x_{ij}(s(j))\right)$$

$$\text{i.e., } \sum_{i \in I} a_{ij} x_{ij}(s(j)) + \sum_{i=1}^{n} a_{ij} x_{ij} > 1$$

$$\text{i.e., } \sum_{i \in I} a_{ij} x_{ij}(s(j)) > \frac{1}{2} \quad \left(\text{since } \sum_{i=1}^{n} x_{ij} < \frac{1}{2} \text{ in Algorithm 12}\right).$$

The desired result now follows from (4.11). □

We now combine these lemmas to show the following result:

Theorem 36. *There is an algorithm for the online covering problem with box constraints that produces an online solution whose cost is at most $O(\log k) \cdot$ DYNAMIC in the multiple predictions setting with k predictions.*

Proof. Initially, let $x_{ij} = y_i = 0$ for all i, j but let $x_{ij}^{\text{DYN}}, y_i^{\text{DYN}}$ be their final value. Then, by Lemma 34, the potential ϕ is at most $O(\log k)$ times the cost of DYNAMIC. Now, as Algorithm 12 increases the values of the variables x_{ij}, y_i, it incurs cost at rate at most $\frac{3}{2}$ (by Lemma 32) and the potential ϕ decreases at rate at least $\frac{1}{2}$ (by Lemma 35). Since ϕ is always non-negative (by Lemma 33), it follows that the total cost of the algorithm is at most 3 times the potential ϕ at the beginning, i.e., at most $O(\log k)$ times the DYNAMIC benchmark. □

We can also robustify the solution produced by Algorithm 12 using the same ideas as in Theorem 26. Namely, we run Algorithm 12 to produce one solution and an online

algoeithm without prediction to obtain another solution. Then, these two solutions are fed into a meta algorithm running Algorithm 12 to obtain the final solution. We get the following analog of Theorem 26:

Theorem 37. *Suppose a class of online covering problems with box constraints have an online algorithm (without predictions) whose competitive ratio is α. Then, there is an algorithm for this class of online covering problems with box constraint with k suggestions that produces an online solution whose cost is at most $O(\min\{\ln k \cdot \text{DYNAMIC}, \alpha \cdot \text{OPT}\})$.*

4.7 Online Facility Location

We now apply the online covering with box constraints framework to the online FACILITY LOCATION problem. An instance of FACILITY LOCATION is given by a set of potential facility locations $\mathscr{F}$, a set $\mathscr{R}$ of client locations and a metric space $d(\cdot,\cdot)$ defined on these points. Further, each location $f_i \in \mathscr{F}$ has a facility opening cost o_i. A solution opens a subset $\mathscr{F}' \subseteq \mathscr{F}$ of facilities, and assigns each client r_j to the closest open facility $f_{i_j} \in \mathscr{F}'$. The connection cost of this client is $d(r_j, f_{i_j})$ and the goal is to minimize the sum of the connection costs of all the clients and the facility opening costs of the facilities in $\mathscr{F}'$.

In the online FACILITY LOCATION problem, clients arrive over time and the solution needs to assign each arriving client to an open facility (possibly after opening a new facility). This problem was first studied by Meyerson (2001) who gave a competitive ratio of $O(\log n)$ for n clients. This was later improved by Fotakis (2008) to $O(\frac{\log n}{\log\log n})$. Since then, many variants of the problem have been studied; for a survey of the results, the reader is referred to Fotakis (2011).

We now encode this problem in the online covering (with box constraints) framework. We have a variable y_i for each facility location $f_i \in \mathscr{F}$, which denotes the extent to which facility f_i is open, and a variable x_{ij} for each client r_j and facility f_i denoting the extent to which r_j is assigned to f_i. Notice that this is a special case of the online covering

problem with box constraints, where $c_i = o_i$ for all i, and $a_{ij} = 1$ for all i, j. As a corollary of Theorem 37, we get the following result:

Theorem 38. *There is an algorithm for the fractional online* FACILITY LOCATION *problem that produces an online solution whose cost is at most* $O(\min\left\{\frac{\log n}{\log\log n}\cdot \text{OPT}, \log k \cdot \text{DYNAMIC}\right\})$ *in the multiple predictions setting with k predictions.*

We also note that the $O(\log k)$ bound in Theorem 38 cannot be improved further (except lower order terms). This is because we can simulate an instance of the online facility location problem without predictions by giving all prior client locations as suggested facility locations for $k = n$. Furthermore, a lower bound of $\Omega\left(\frac{\log n}{\log\log n}\right)$ is also known (due to Fotakis Fotakis (2008)) for online facility location, and this lower bound construction easily extends to the fractional version of the problem.

4.8 Future Directions

This chapter presented a general recipe for the design of online algorithms with multiple machine-learned predictions. This general technique was employed to obtain tight results for classical online problems such as set cover, (weighted) caching, and facility location. We believe this framework can be applied to other online covering problems as well. In particular, it would be interesting to consider the k-server problem with multiple suggestions in each step specifying the server that should serve the new request. It would also be interesting to extend this framework to some packing problems such as online budgeted allocation, and to more general settings for mixed packing and covering linear programs and non-linear (convex) objectives. From a technical standpoint, our potential-based analysis introduces several new ideas that can be useful even in the competitive analysis of classical online algorithms.

4.9 Chapter Notes

The work presented in this chapter was done jointly with Rong Ge, Amit Kumar and Debmalya Panigrahi. A preliminary version is going to appear in International Conference of Machine Learning (ICML), 2022. (see Anand et al. (2022))

5

Concluding Remarks

In this thesis, we studied how pessimistic worst-case bounds of traditional online algorithms can be circumvented by designing and analysing online algorithms with learned predictions. We examined how being cognizant of the design and optimization benchmarks of the online problem can help in designing a better prediction pipeline. The forecasts of such pipelines are synergistic with the goals of the online decision maker and can vastly improve its performance. This general approach of "learning for optimization" opens up a new direction for future research at the boundary of machine learning and algorithm design, by providing an alternative "white box" approach to the existing "black box" approaches for using ML predictions in *beyond worst case* algorithm design. While we explored this for an online problem in this chapter, the principle itself can be applied to any scenario where an algorithm hopes to learn patterns in the input that can be exploited to achieve performance gains. We posit that this is a rich direction for future research.

We also examined a setting where multiple predictions are provided to the decision maker and the task is to perform just as well as the best set of predictions. For the multiple prediction setting, an interesting direction of future research would be try to extend the

result in the Metrical Task Systems (MTS) setting.

Bibliography

Achlioptas, D., Chrobak, M., and Noga, J. (2000), "Competitive analysis of randomized paging algorithms," *Theor. Comput. Sci.*, 234, 203–218.

Ai, L., Wu, X., Huang, L., Huang, L., Tang, P., and Li, J. (2014), "The multi-shop ski rental problem," in *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '14, Austin, TX, USA - June 16 - 20, 2014*, eds. S. Sanghavi, S. Shakkottai, M. Lelarge, and B. Schroeder, pp. 463–475, ACM.

Albers, S. (1999), "Better bounds for online scheduling," *SIAM Journal on Computing*, 29, 459–473.

Almanza, M., Chierichetti, F., Lattanzi, S., Panconesi, A., and Re, G. (2021), "Online Facility Location with Multiple Advice," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. (2006), "A general approach to online network optimization problems," *ACM Transactions on Algorithms*, 2, 640–660.

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. (2009), "The Online Set Cover Problem," *SIAM J. Comput.*, 39, 361–370.

Anand, K., Ge, R., and Panigrahi, D. (2020), "Customizing ML Predictions for Online Algorithms," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 303–313, PMLR.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. (2021), "A Regression Approach to Learning-Augmented Online Algorithms," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. (2022), "Online Algorithms with Multiple Predictions," *arXiv preprint arXiv:2205.03921*.

Angelopoulos, S., Dürr, C., Kamali, S., Renault, M. P., and Rosén, A. (2015), "Online Bin Packing with Advice of Small Size," in *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, eds. F. Dehne, J. Sack, and U. Stege, vol. 9214 of *Lecture Notes in Computer Science*, pp. 40–53, Springer.

Anthony, M. M. and Bartlett, P. (1999), *Learning in neural networks: theoretical foundations*, Cambridge University Press.

Awasthi, P., Balcan, M. F., and Long, P. M. (2014), "The power of localization for efficiently learning linear separators with noise," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 449–458, ACM.

Azar, Y. and Regev, O. (1998), "On-Line Bin-Stretching," in *Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings*, eds. M. Luby, J. D. P. Rolim, and M. J. Serna, vol. 1518 of *Lecture Notes in Computer Science*, pp. 71–81, Springer.

Azar, Y., Bhaskar, U., Fleischer, L. K., and Panigrahi, D. (2013), "Online Mixed Packing and Covering," in *SODA*.

Azar, Y., Buchbinder, N., Chan, T. H., Chen, S., Cohen, I. R., Gupta, A., Huang, Z., Kang, N., Nagarajan, V., Naor, J., and Panigrahi, D. (2016), "Online Algorithms for Covering and Packing Problems with Convex Objectives," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 148–157.

Bamas, É., Maggiori, A., and Svensson, O. (2020), "The Primal-Dual method for Learning Augmented Algorithms," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin.

Bansal, N. and Coester, C. (2021), "Online metric allocation," *CoRR*, abs/2111.15169.

Bansal, N., Buchbinder, N., and Naor, J. (2007), "A Primal-Dual Randomized Algorithm for Weighted Paging," in *FOCS*, pp. 507–517.

Bansal, N., Buchbinder, N., and Naor, J. S. (2010), "A simple analysis for randomized online weighted paging," *Unpublished Manuscript*.

Bartal, Y., Fiat, A., Karloff, H. J., and Vohra, R. (1992), "New Algorithms for an Ancient Scheduling Problem," in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, eds. S. R. Kosaraju, M. Fellows, A. Wigderson, and J. A. Ellis, pp. 51–58, ACM.

Bhaskara, A., Cutkosky, A., Kumar, R., and Purohit, M. (2020), "Online linear optimization with many hints," *Advances in neural information processing systems*, 33, 9530–9539.

Blum, A., Frieze, A., Kannan, R., and Vempala, S. (1998), "A polynomial-time algorithm for learning noisy linear threshold functions," *Algorithmica*, 22, 35–52.

Blum, A., Burch, C., and Kalai, A. (1999), "Finely-Competitive Paging," in *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pp. 450–458, IEEE Computer Society.

Böhm, M., Sgall, J., van Stee, R., and Veselý, P. (2017), "A two-phase algorithm for bin stretching with stretching factor 1.5," *J. Comb. Optim.*, 34, 810–828.

Borodin, A. and El-Yaniv, R. (1998), *Online Computation and Competitive Analysis*, Cambridge University Press.

Buchbinder, N. and Naor, J. (2009a), "The Design of Competitive Online Algorithms via a Primal-Dual Approach," *Foundations and Trends in Theoretical Computer Science*, 3, 93–263.

Buchbinder, N. and Naor, J. (2009b), "Online Primal-Dual Algorithms for Covering and Packing," *Math. Oper. Res.*, 34, 270–286.

Buchbinder, N., Jain, K., and Naor, J. (2007), "Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue," in *ESA*, pp. 253–264.

Buchbinder, N., Gupta, A., Molinaro, M., and Naor, J. S. (2019), "k-Servers with a Smile: Online Algorithms via Projections," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, ed. T. M. Chan, pp. 98–116, SIAM.

Bylander, T. (1994), "Learning linear threshold functions in the presence of classification noise," in *Proceedings of the seventh annual conference on Computational learning theory*, pp. 340–347.

Chrobak, M., Karloof, H., Payne, T., and Vishwnathan, S. (1991), "New results on server problems," *SIAM Journal on Discrete Mathematics*, 4, 172–181.

Craig, C. C. (1933), "On the Tchebychef inequality of Bernstein," *The Annals of Mathematical Statistics*, 4, 94–102.

Fiat, A., Karp, R. M., Luby, M., McGeoch, L. A., Sleator, D. D., and Young, N. E. (1991), "Competitive Paging Algorithms," *J. Algorithms*, 12, 685–699.

Fleischer, R. (2001), "On the Bahncard problem," *Theor. Comput. Sci.*, 268, 161–174.

Fleischer, R. and Wahl, M. (2000), "Online Scheduling Revisited," in *Algorithms - ESA 2000, 8th Annual European Symposium, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, ed. M. Paterson, vol. 1879 of *Lecture Notes in Computer Science*, pp. 202–210, Springer.

Fotakis, D. (2008), "On the Competitive Ratio for Online Facility Location," *Algorithmica*, 50, 1–57.

Fotakis, D. (2011), "Online and incremental algorithms for facility location," *SIGACT News*, 42, 97–131.

Gabay, M., Kotov, V., and Brauner, N. (2015), "Online bin stretching with bunch techniques," *Theor. Comput. Sci.*, 602, 103–113.

Gabay, M., Brauner, N., and Kotov, V. (2017), "Improved lower bounds for the online bin stretching problem," *4OR*, 15, 183–199.

Galambos, G. and Woeginger, G. J. (1993), "An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling," *SIAM Journal on Computing*, 22, 349–355.

Gollapudi, S. and Panigrahi, D. (2019a), "Online Algorithms for Rent-Or-Buy with Expert Advice," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 2319–2327.

Gollapudi, S. and Panigrahi, D. (2019b), "Online Algorithms for Rent-Or-Buy with Expert Advice," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, eds. K. Chaudhuri and R. Salakhutdinov, vol. 97 of *Proceedings of Machine Learning Research*, pp. 2319–2327, PMLR.

Gormley, T., Reingold, N., Torng, E., and Westbrook, J. (2000), "Generating adversaries for request-answer games," in *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 564–565.

Graham, R. L. (1969), "Bounds on Multiprocessing Timing Anomalies," *SIAM Journal of Applied Mathematics*, 17, 416–429.

Gupta, A. and Nagarajan, V. (2014), "Approximating Sparse Covering Integer Programs Online," *Math. Oper. Res.*, 39, 998–1011.

Hoeffding (1963), "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, pp. 13–30.

Karger, D. R., Phillips, S. J., and Torng, E. (1996), "A better algorithm for an ancient scheduling problem," *Journal of Algorithms*, 20, 400–430.

Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. (1988), "Competitive snoopy caching," *Algorithmica*, 3, 77–119.

Karlin, A. R., Manasse, M. S., McGeoch, L. A., and Owicki, S. (1994), "Competitive randomized algorithms for nonuniform problems," *Algorithmica*, 11, 542–571.

Karlin, A. R., Kenyon, C., and Randall, D. (2003), "Dynamic TCP Acknowledgment and Other Stories about e/(e-1)," *Algorithmica*, 36, 209–224.

Kearns, M. J. and Vazirani, U. V. (1994), *An introduction to computational learning theory*, MIT press.

Kellerer, H. and Kotov, V. (2013), "An efficient algorithm for bin stretching," *Oper. Res. Lett.*, 41, 343–346.

Khanafer, A., Kodialam, M., and Puttaswamy, K. P. N. (2013), "The constrained ski-rental problem and its application to online cloud cost optimization," in *Proceedings of the INFOCOM*, pp. 1492–1500.

Kodialam, R. (2014), "Competitive algorithms for an online rent or buy problem with variable demand," *SIAM Undergraduate Research Online*, 7, 233–245.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. (2020), "Online Scheduling via Learned Weights," in *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, ed. S. Chawla, pp. 1859–1877, SIAM.

Lavastida, T., Moseley, B., Ravi, R., and Xu, C. (2020), "Learnable and Instance-Robust Predictions for Online Matching, Flows and Load Balancing," *arXiv preprint arXiv:2011.11743*.

Lotker, Z., Patt-Shamir, B., and Rawitz, D. (2008), "Rent, Lease or Buy: Randomized Algorithms for Multislope Ski Rental," in *Proceedings of the 25th Annual Symposium on the Theoretical Aspects of Computer Science (STACS)*, pp. 503–514.

Lotker, Z., Patt-Shamir, B., and Rawitz, D. (2012), "Rent, Lease, or Buy: Randomized Algorithms for Multislope Ski Rental," *SIAM J. Discret. Math.*, 26, 718–736.

McGeoch, L. A. and Sleator, D. D. (1991), "A Strongly Competitive Randomized Paging Algorithm," *Algorithmica*, 6, 816–825.

Meyerson, A. (2001), "Online Facility Location," in *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pp. 426–431, IEEE Computer Society.

Meyerson, A. (2005a), "The Parking Permit Problem," in *Proc. of 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 274–284.

Meyerson, A. (2005b), "The Parking Permit Problem," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pp. 274–284, IEEE Computer Society.

Motwani, R. and Raghavan, P. (1997), *Randomized Algorithms*, Cambridge University Press.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013), "Learning with noisy labels," in *Advances in neural information processing systems*, pp. 1196–1204.

Purohit, M., Svitkina, Z., and Kumar, R. (2018), "Improving online algorithms via ml predictions," in *Advances in Neural Information Processing Systems*, pp. 9661–9670.

Rasmussen, C. E. (2003), "Gaussian processes in machine learning," in *Summer School on Machine Learning*, pp. 63–71, Springer.

Shalev-Shwartz, S. (2012), "Online Learning and Online Convex Optimization," *Found. Trends Mach. Learn.*, 4, 107–194.

Sleator, D. D. and Tarjan, R. E. (1985), "Amortized Efficiency of List Update and Paging Rules," *Commun. ACM*, 28, 202–208.

Vapnik, V. and Vapnik, V. (1998), "Statistical learning theory Wiley," *New York*, pp. 156–160.

Wang, S., Li, J., and Wang, S. (2020), "Online Algorithms for Multi-shop Ski Rental with Machine Learned Advice," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin.

Yao, A. C.-C. (1977), "Probabilistic computations: Toward a unified measure of complexity," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 222–227, IEEE Computer Society.

Young, N. (1991), "On-line Caching As Cache Size Varies," in *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '91, pp. 241–250, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics.

Young, N. E. (1994), "The k-Server Dual and Loose Competitiveness for Paging," *Algorithmica*, 11, 525–541.