Foundation of Machine Learning, by the People, for the People

Nika Haghtalab

CMU-CS-18-114

August 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Avrim Blum, Co-chair
Ariel D. Procaccia, Co-chair
Maria-Florina Balcan, Carnegie Mellon University
Tim Roughgarden, Stanford University
Robert Schapire, Microsoft Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by an IBM Ph.D. Fellowship, a Microsoft Research Fellowship, a Siebel Scholarship, and the National Science Foundation under grant numbers IIS-1065251, IIS-1350598, CCF-1451177, CCF-1525971, and CCF-1535967. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Machine learning, Algorithmic economics, Theory of Computer Science, Mechanism design, Stackelberg Games, Auction Design, No-regret Learning, Collaborative learning, Learning from the crowd, Kidney Exchange

To the Iranian Bahá'í Community.

Abstract

Typical analysis of *machine learning* algorithms considers their outcome in isolation from the effects that they may have on the process that generates the data or the entity that is interested in learning. However, current technological trends mean that people and organizations increasingly interact with learning systems, making it necessary to consider how these interactions change the nature and outcome of learning tasks.

The field of *algorithmic game theory* has been developed in response to the need for understanding interactions in large interactive systems in the presence of strategic entities, such as people. In many cases, however, algorithmic game theory requires an accurate model of people's behavior. In the applications of machine learning, however, much of this information is unavailable or evolving. So, in addition to the challenges involved in algorithmic game theory, there is a need to acquire the information without causing undesirable interactions.

In this thesis, we present a view of machine learning and algorithmic game theory that considers the interactions between machine learning systems and people. We explore four lines of research that account for these interactions: *learning about people*, where we learn optimal policies in game-theoretic settings, without an accurate behavioral model and in ever changing environments, by interacting with and learning about people's preferences; *learning from people*, where we manage people's expertise and resources in data-collection and machine learning; *learning by people*, where people can interact with each other and collaborate together to effectively learn related underlying concepts; and *learning for people*, where machine learning is used to benefit people and society, in particular, by creating models that are resilient to uncertainties in the environment.

Acknowledgments

I have been lucky to have not just one but two amazing advisors! In addition to being fantastic researchers, Avrim and Ariel are two of the kindest and most supportive individuals that I have had the pleasure of knowing. Avrim has been a great teacher with an incredibly intuitive understanding of the theory of computer science and superb technical skills. I want to thank him for his encouragements and his insights, for sharing with me his knowledge of a wide range of research, for his ability to form elegant questions so that their answer emerges with equal elegance, and for his humility and patience, which enabled me to grow as a researcher and come into my own. Ariel has been a great friend and mentor. He has a great sense of style in choosing what to work on. His approach and interests are so novel that people who cross his path cannot help but want to work with him. I want to thank him for his friendship and his time, for his sense of worthwhile research areas, for his ability to provide a computational perspective on all of life's problems, for his passion for perfection that demonstrates itself in all aspects of his research, and for his unwavering support and faith in me that pushes me to strive to be a better researcher.

I am grateful to Nina Balcan, Tim Roughgarden, and Rob Schapire for all that they have done for me during my Ph.D., including serving on my thesis committee. Nina has been a collaborator on a number of projects discussed in this thesis. I want to thank her for her transformative energy that has influenced much of my taste in problems. Tim hosted me for a visit at Stanford, the results of which are presented in Chapter 7 of this thesis. During this time, he helped me achieve a better understanding of the wider range of connections between machine learning and the theory of computer science. Rob was one of my hosts during a very fun and productive internship at Microsoft Research, the results of which are presented in Chapter 5 of this thesis. At times when our goals seemed unattainable, Rob's calmness and faith helped me keep my focus and rigor.

A huge thanks goes to all of my other collaborators throughout my Ph.D: Nima Anari, Pranjal Awasthi, Ioannis Caragiannis, Ofer Dekel, John dickerson, Miro Dudìk, Fei Fang, Arthur Flajolet, Patrick Jaillet, Aron Laszka, Haipeng Luo, Simon MacKenzie, Yishay Mansour, Seffi Naor, Thanh Ngyuen, Ritesh Noothigattu, Sebastian Pokutta, Eviatar Procaccia, Mingda Qiao, Oren Salzman, Tuomas Sandholm, Ankit Sharma, Mohit Singh, Arunesh Sinha, Sid Srinivasa, Vasilis Sygkanis, Alfredo Torrico, Milind Tambe, Ruth Urner, Rohit Vaish, Yevgeniy Vorobeychik, Colin White, Jenn Wortman Vaughan, and Hongyang Zhang. It would not have been as fun or as productive without them. Especially, I want to thank Ofer Dekel, Miro Dudìk, Jenn Wortman Vaughan, Rob Schapire, and Vasilis Syrgkanis for being amazing mentors during two very fun summers at Microsoft Research Redmond and New York City.

I want to thank everyone at CMU for contributing to a great environment for graduate studies.

I am thankful to the members of the theory group, and especially to Mor Harchol-Balter and Anupam Gupta, for their company and advice. I also want to thank Deb Cavlovich, Catherine Copetas, Patricia Loring, and other amazing administrative staff for making the everyday life at CMU so easy for graduate students. Special thanks to all of my friends and peers that made my Ph.D. years some of the most memorable years of my life. I cannot possible name all of them; instead let me thank them for sharing their love, thoughts, time, advice, houses, happiness, sadness, and coffee/tea with me!

Lastly, I am ever indebted to my family — my parents Felora and Nasser, my sister Ayda, and my husband Erik — for having my back, being my unabashed champions, and creating in me a thirst for learning and education. They hold me to a high standard and help me achieve it. They have done so much for me before my Ph.D. and I know that they will continue to do so much for me after my Ph.D. that nothing I can say would sufficiently convey my love and appreciation for them.

Contents

1 Introduction **1**
- 1.1 Background . . . 3
  - 1.1.1 Stackelberg Games . . . 3
  - 1.1.2 Offline Learning . . . 4
  - 1.1.3 Online Learning . . . 5
- 1.2 Overview of Thesis Contributions and Structure . . . 7
- 1.3 Bibliographical Remarks . . . 18
- 1.4 Excluded Research . . . 19

I Learning about People **21**

2 Learning in Stackelberg Security Games **23**
- 2.1 Introduction . . . 23
- 2.2 The Model . . . 24
- 2.3 Problem Formulation and Technical Approach . . . 26
- 2.4 Main Result . . . 27
  - 2.4.1 Characteristics of the Optimization Region . . . 27
  - 2.4.2 Finding Initial Points . . . 29
  - 2.4.3 An Oracle for the Convex Region . . . 33
  - 2.4.4 The Algorithms . . . 33
- 2.5 Discussion . . . 36

3 Learning about a Boundedly Rational Attacker in Stackelberg Games **39**
- 3.1 Introduction . . . 39
  - 3.1.1 Our Results . . . 39
  - 3.1.2 Related Work . . . 40
- 3.2 Preliminaries . . . 41
- 3.3 Theoretical Results . . . 42
  - 3.3.1 Linear Utility Functions . . . 42
  - 3.3.2 Polynomial Utility Functions . . . 44
  - 3.3.3 Lipschitz Utilities . . . 46
  - 3.3.4 Learning the Optimal Strategy . . . 47
- 3.4 Discussion and Open Problems . . . 48

4 Online Learning in Multi-attacker Stackelberg Games **51**
4.1 Introduction . . . 51
4.1.1 Overview of Our Results . . . 52
4.1.2 Related work . . . 52
4.2 Preliminaries . . . 53
4.3 Problem Formulation . . . 54
4.3.1 Methodology . . . 55
4.4 Characteristics of the Offline Optimum . . . 56
4.5 Upper bounds – Full Information . . . 58
4.6 Upper bounds – Partial Information . . . 59
4.6.1 Overview of the Approach . . . 60
4.6.2 Partial Information to Full Information . . . 61
4.6.3 Creating Unbiased Estimators . . . 63
4.6.4 Putting It All Together . . . 65
4.7 Lower Bound . . . 66
4.8 Discussion . . . 69
4.9 Subsequent Works . . . 70

5 Oracle-Efficient Online Learning and Auction Design **71**
5.1 Introduction . . . 71
5.1.1 Oracle-Efficient Learning with Generalized FTPL . . . 73
5.1.2 Main Application: Online Auction Design . . . 75
5.1.3 Extensions and Additional Applications . . . 77
5.2 Generalized FTPL and Oracle-Efficient Online Learning . . . 78
5.2.1 Regret Analysis . . . 80
5.2.2 Oracle-Efficient Online Learning . . . 82
5.3 Online Auction Design . . . 85
5.3.1 VCG with Bidder-Specific Reserves . . . 86
5.3.2 Envy-free Item Pricing . . . 91
5.3.3 Level Auctions . . . 93
5.4 Stochastic Adversaries and Stronger Benchmarks . . . 95
5.4.1 Stochastic Adversaries . . . 95
5.4.2 Implications for Online Optimal Auction Design . . . 97
5.5 Approximate Oracles and Approximate Regret . . . 101
5.5.1 Approximation through Relaxation . . . 102
5.5.2 Approximation by Maximal-in-Range Algorithms . . . 103
5.6 Additional Applications and Connections . . . 103
5.6.1 Fully Efficient Online Welfare Maximization in Multi-Unit Auctions . . . 103
5.6.2 Oracle Efficient Online Bidding in Simultaneous Second Price Auctions 105
5.6.3 Universal Identification Sequences . . . 107

6 Online Learning with a Hint **109**
6.1 Introduction . . . 109
6.2 Related work . . . 110
6.3 Preliminaries . . . 111
6.4 Improved Regret Bounds for Strongly Convex $\mathcal{K}$ . . . 112
6.5 Improved Regret Bounds for (C, q)-Uniformly Convex $\mathcal{K}$ . . . 115
6.6 Lack of uniform Convexity . . . 119
6.7 Discussion . . . 121
6.7.1 Comparison with other Notions of Hint . . . 121

7 Smoothed Analysis of Online Learning **123**
7.1 Introduction . . . 123
7.1.1 Smoothed Analysis . . . 123
7.1.2 Smoothed Analysis in Online Learning . . . 124
7.1.3 Our Results . . . 125
7.1.4 Related Work . . . 125
7.2 Preliminaries . . . 126
7.3 Main Results . . . 127
7.4 Lower Bound for Non-adaptive Non-smooth Adversaries . . . 130
7.5 Discussion and Open Problem . . . 131
7.5.1 An Open Problem . . . 132

II Learning from People **135**

8 Learning with Bounded Noise **137**
8.1 Introduction . . . 137
8.1.1 Our Results . . . 138
8.1.2 Our Techniques . . . 139
8.1.3 Related Work . . . 140
8.2 Preliminaries . . . 142
8.3 Bounded Noise Algorithm . . . 143
8.3.1 Outline of the Proof and Related Lemmas . . . 145
8.3.2 Initializing $\mathbf{w}_0$ . . . 148
8.3.3 Putting Everything Together . . . 149
8.4 AVERAGE Does Not Work . . . 150
8.5 Hinge Loss Minimization Does Not Work . . . 152
8.5.1 Proof of the Lower Bound . . . 153
8.6 Discussion and Subsequent Works . . . 156
8.6.1 Subsequent Works . . . 157

9 Efficient PAC Learning from the Crowd **159**
9.1 Introduction . . . 159
9.1.1 Overview of Results . . . 160

9.1.2 Related Work . . . 162
9.2 Model and Notations . . . 163
9.3 A Baseline Algorithm and a Road-map for Improvement . . . 165
9.4 An Interleaving Algorithm . . . 165
9.4.1 The General Case of Any α . . . 173
9.5 No Perfect Labelers . . . 177

III Learning by People **181**

10 Collaborative PAC Learning **183**
10.1 Introduction . . . 183
10.1.1 Overview of Results . . . 184
10.1.2 Related Work . . . 184
10.2 Model . . . 185
10.3 Sample Complexity Upper Bounds . . . 186
10.3.1 Personalized Setting . . . 186
10.3.2 Centralized Setting . . . 188
10.4 Sample Complexity Lower Bounds . . . 191
10.4.1 Tight Lower Bound for the Personalized Setting . . . 192
10.4.2 Lower Bound for Uniform Convergence . . . 195
10.5 Extension to the Non-realizable Setting . . . 196
10.6 Discussion and Subsequent Works . . . 198

IV Learning for People **199**

11 A Near Optimal Kidney Exchange with a Few Queries **201**
11.1 Introduction . . . 201
11.1.1 Our theoretical results and techniques . . . 202
11.1.2 Our experimental results: Application to kidney exchange . . . 203
11.2 Related work . . . 204
11.2.1 Stochastic matching . . . 205
11.2.2 Kidney exchange . . . 205
11.2.3 Subsequent Work . . . 206
11.3 The Model . . . 206
11.4 Understanding the Challenges . . . 207
11.5 Adaptive Algorithm: $(1-\epsilon)$-approximation . . . 208
11.6 Non-adaptive algorithm: 0.5-approximation . . . 211
11.6.1 Upper Bound on the Performance of the Non-Adaptive Algorithm . . . 213
11.7 Generalization to stochastic k-cycle packing . . . 215
11.7.1 Augmenting structures for k-cycle packing . . . 216
11.7.2 Adaptive algorithm for k-set packing . . . 218
11.8 Experimental Results . . . 220

11.8.1 Experiments on dense generated graphs . . . 221
11.8.2 Experiments on real match runs from the UNOS nationwide kidney exchange . . . 222
11.9 Discussion & future research . . . 225
11.9.1 Open theoretical problems . . . 226
11.9.2 Discussion of policy implications of experimental results . . . 227

12 Individually Rational Multi-Hospital Kidney Exchange 229
12.1 Introduction . . . 229
12.1.1 Our Approach . . . 230
12.1.2 Our Results and Techniques . . . 230
12.1.3 Related Work . . . 232
12.2 Optimal Matchings Are Almost Individually Rational . . . 233
12.2.1 Proof of Lemma 12.2.3 . . . 235
12.2.2 Proof of Lemma 12.2.4 . . . 239
12.3 Individually Rational Matchings that Are Almost Optimal . . . 242
12.4 Justification for Conditions on p and L . . . 245
12.4.1 The Case of Large p . . . 246
12.4.2 The Case of Long Cycles . . . 246
12.5 Conclusions and Open Problems . . . 247

A Omitted Proofs for Chapter 5 249
A.1 Proof of Equation 5.4 . . . 249
A.2 Proof of Lemma 5.2.1 . . . 250
A.3 Proof of Lemma 5.3.9 . . . 251
A.4 Proof of Lemma 5.4.1 . . . 253
A.5 Proof of Lemma 5.4.3 . . . 253

B Omitted Proofs of Chapter 8 255
B.1 Proof of Lemma 8.5.2 . . . 255
B.2 Proof of Lemma 8.5.3 . . . 256
B.3 Proof of Lemma 8.5.4 . . . 256

C Probability Lemmas for Chapter 9 259

Bibliography 261

List of Figures

1.1 A machine learning framework that accounts for strategic and social interactions. 2

1.2 The figure on the left demonstrates instances labeled black and white that are perfectly classified by a halfspace. The figure in the middle demonstrates adversarial noise, where the adversary deterministically flips the labels of 10% of the data (the shaded region). The figure on the right demonstrates the random classification noise where the labels of all points (the shaded area) are flipped with probability 10%. 14

2.1 The game payoff table on the right and optimization regions on the left. A security game with one resource that can cover one of two targets. The attacker receives utility 0.5 from attacking target 1 and utility 1 from attacking target 2, when they are not defended; he receives 0 utility from attacking a target that is being defended. The defender's utility is the zero-sum complement. 30

4.1 Best-response regions. The first two figures define $\mathcal{P}_i^j$ in a game where one resource can cover one of two targets, and two attacker types. The third figure illustrates $\mathcal{P}_\sigma$ for the intersection of the best-response regions of the two attackers. 57

5.1 $\mathbf{\Gamma}^{\mathrm{VCG}}$ for $n = 2$ bidders and $m = 3$ 88

5.2 Demonstration of how $\boldsymbol{\theta}$ can be reconstructed by its revenue on the bid profiles in $V = \{\mathbf{v}^{i,\ell}\}_{i,\ell} \cup \{\mathbf{e}_n\}$. On the left, we show that as the value v_n (blue circle) gradually increases from 0 to 1, the revenue of the auction (red vertical lines) jumps along the sequence of values $\theta_0^i, \theta_1^i, \ldots, \theta_{s-1}^i$. So by analyzing the revenue of an auction on all bid profiles $\{\mathbf{v}^{i,\ell}\}_{i,\ell}$ one can reconstruct $\boldsymbol{\theta}^i$ for $i \neq n$ and $\theta_1^n, \ldots, \theta_{s-1}^n$. To reconstruct θ_0^n, one only needs to consider the profile $\mathbf{e}_n$. The figure on the right demonstrates the revenue of the same auction, where the horizontal axis is the value of v_n and the vertical axis is the revenue of the auction when $v_i = 1$ and all other valuations are 0. 94

5.3 Demonstrating cases 1 and 2 of prof of Lemma 5.3.12. The bidder valuations are demonstrated by blue circles on the real line and the revenue of the two auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are demonstrated by red solid vertical line. 94

6.1 Virtual function and its properties. 113

7.1 Path $(+1, -1, -1)$ is associated with the sequence $(\frac{1}{2}, +)$, $(\frac{1}{4}, -)$, and $(\frac{3}{8}, -)$. . 132

8.1 $\mathcal{D}_{\alpha,\beta}$ . . . 153
8.2 Area C . . . 155

11.1 Compatibility graphs for pairwise and three-way exchanges. Solid blue edges represent successful crossmatch tests, dashed blue edges represent failed crossmatch tests, and black edges represent potential compatibilities that have not been tested. Note that when pairwise exchanges are considered, the number of incoming edge tests of a node is the same as the number of its outgoing edge tests—a patient and its willing but incompatible donor are always involved in an equal number of tests—while in three-way exchanges the number of incoming and outgoing edge tests may be different. . . . 204
11.2 Illustration of the construction in Example 11.4.2, for $t = 4$ and $\beta = 1/2$. . . . 208
11.3 Illustration of the upper bound on the performance of non-adaptive algorithm. Blue and red edges represent the matching picked at rounds 1 and 2, respectively. The green edges represent the edges picked at round 3 and above. The dashed edges are never picked by the algorithm. . . . 214
11.4 Saidman generator graphs constrained to 2-cycles only (left) and both 2- and 3-cycles (right). . . . 222
11.5 Real UNOS match runs constrained to 2-cycles (left) and both 2-cycles and chains (right). . . . 223
11.6 Real UNOS match runs with 2- and 3-cycles and no chains (left) and with chains (right). . . . 224
11.7 Real UNOS match runs, restricted matching of 2-cycles only, without chains (left) and with chains (right), including zero-sized omnsicient matchings. . . . 224
11.8 Real UNOS match runs, matching with 2- and 3-cycles, without chains (left) and with chains (right), including zero-sized omnsicient matchings. . . . 225

12.1 A compatibility graph where individual rationality fails. . . . 229
12.2 A graph demonstrating the Edmonds-Gallai Decomposition and the edge-disjoint graph partition for the proof of Lemma 12.2.3. In this graph, each color represents one G_i in the partition $G = \biguplus_i G_i$ and the wavy edges represent the matched edges in $\textsc{Opt}(G)$. . . 237
12.3 The graph construction of Example 12.3.2. . . . 245
12.4 A graph demonstrating the problem with long cycles. . . . 246

B.1 Area T . . . 257

List of Tables

1.1 Applications of online learning to linear optimiation, Stackelberg games, auctions, and prediction. 6

5.1 Regret bounds and oracle-based computational efficiency, for the auction classes considered in this work for n bidders and time horizon T. All our results perform a single oracle call per iteration. 77

5.2 Additional results considered in Sections 5.4-5.6 and their significance. Above, m is the discretization level of the problems, n is the number of bidders, and T is the time horizon. 78

Chapter 1

Introduction

It is no secret that machine learning has had many successes in the real world; it has revolutionized scientific fields, technology, and our day-to-day lives broadly. Progress in machine learning has in part enabled breakthroughs in a variety of applications, such as natural language processing [10, 75, 84, 200, 244], computer vision [102, 175, 175, 192], bioinformatics [40, 187], robotics [1, 135, 191], to name a few. On the theoretical front, elegant and powerful tools in machine learning, such as VC-dimension, Rademacher theory, regret bounds, boosting, etc., have led to deeper understanding of other mathematical fields, such as control theory [111, 120], algorithms [39, 139] and more.

From the theoretical perspective, one of the fundamental questions that the field of machine learning seeks to answer is how to design and analyze algorithms that compute general facts about an underlying data-generating process by observing a limited amount of that data. At a high level, this question suggests viewing machine learning as a framework with three building blocks: 1) an unknown process that generates data, 2) a process that collects limited amount of that data, and 3) a learner who is interested in learning some general facts about the former by studying the latter. As an example, when considering passive supervised learning in this framework, we have an unknown distribution $\mathcal{D}$ (the data-generating process) over instances $\mathcal{X}$ labeled $+1$ and -1, a sample set of instances chosen i.i.d. from $\mathcal{D}$ (the data-collection process), and a learner who is interested in finding within a pre-determined set of functions $\mathcal{H}$ a function $h \in \mathcal{H}$ (the general fact) that best describes how instances in $\mathcal{D}$ map to labels $+1$ and -1.

Traditionally the outcome of a learning algorithm has been considered in isolation from the effects that it may have on the process that generates the data or the entity who is interested in learning. With data science and the applications of machine learning revolutionizing day-to-day life, however, increasingly more people and organizations interact with learning systems. In their interactions, people and organizations demonstrate a wide range of social and economic limitations, aspirations, and behaviors. This necessitates a deeper understanding of how these interactions fundamentally change the nature and outcome of the learning tasks and the challenges involved.

The field of *algorithmic game theory* has been developed in response to the need for understanding interactions in large interactive systems. Being positioned at the intersection of Computer Science, Economics, and Game Theory, this field answers questions such as: *what kind of behavior emerges when people make selfish decisions or selflessly collaborate [185, 236, 238]?*

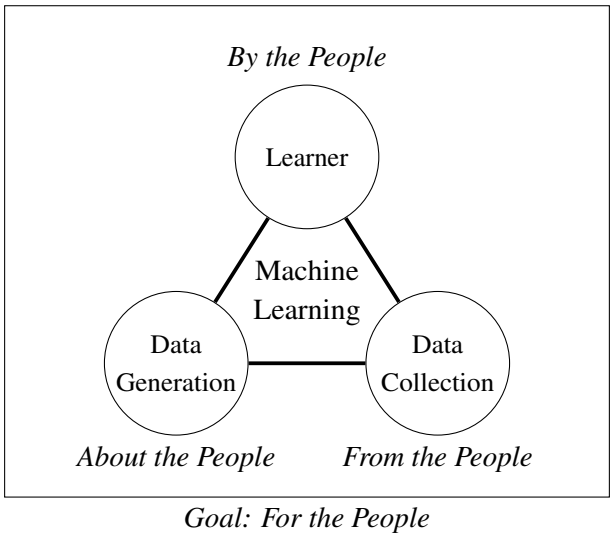


Figure 1.1: A machine learning framework that accounts for strategic and social interactions.

How can we compute the outcome (equilibria) of these interactions [28, 93, 95, 218, 241]? How can we design mechanisms that incentivize certain types of behavior [91, 134, 162, 188, 212, 215]?

While many attribute the development of the field of algorithmic game theory to the advent of the Internet—one of the largest computational systems that has emerged from the strategic interactions of many entities—participation of people and organizations in machine learning systems has introduced interesting and novel challenges in the intersection of algorithmic game theory and machine learning. In many cases, algorithmic game theory requires significant amount of information about the players in terms of an accurate model of their behavior. In the applications of machine learning, however, much of this information is unavailable or evolving. So, in addition to computational and analytical challenges involved in algorithmic game theory, there are statistical challenges on how to efficiently gain the information that algorithmic game theory relies on without causing undesirable interactions.

In this thesis we advocate a view of machine learning and algorithmic game theory that considers the interactions between machine learning systems and people. Broadly speaking, these interactions include settings where ...

(About the People) the process that generates the data is a group of people or organizations,

(From the People) data is collected by people or organizations, i.e., learning from the crowd,

(By the People) the learning task is performed by a person or organization,

(For the People) the overarching goal of the system is to benefit people and the society.

To see how interactions with people and organizations change the way one should design learning systems, consider the following example: suppose Whole Foods wishes to decide what items should be offered at one of its branches and at what prices. To do so, Whole Foods has to

develop a good understanding of customer preferences (learn about people). This information is seldom available freely. However, customers' interactions with the current pricing scheme, in the form of their purchases, reveal important information about their preferences. Here, Whole Foods has the opportunity to learn and refine its pricing scheme by using these interactions. Moreover, customer preferences may evolve over time as a result of their earlier interactions with the mechanism. For example, buyers who have recently purchased a one-year supply of an item may not be interested in the same item, even at a deep discount, in the near future. Therefore, the learning process should account for the social and economic interactions between the customers and the mechanism. Suppose that Whole Foods decides to learn about the preferences of the community by surveying a few people (learning from people). An individual has limited time and interest and may be willing to answer only a few questions. So, a successful learning mechanism should account for this limitation and effectively learn complex facts about the larger community using only a few questions per individual. In some cases, multiple branches of Whole Foods or a competing supermarket may be conducting market research simultaneously (learning by people). In this case, the learning process should account for how interactions with other branches or firms may benefit or harm each firm.

The interactions between people and learning systems exists in many applications and domains. In industry alone, a recent survey [235] estimates that 85 of the 100 "top global brands" either directly or indirectly use human knowledge and behavior in designing better products. In social causes, learning in presence of interactions with people has made a big impact on how we go about preserving wildlife [116, 142], securing our cities [256], and has been used for creating better and more effective disaster relief systems [132, 217]. In science and education, Massive Open Online Courses rely on machine learning to create better environments for people to learn and interact [221]. The importance and prevalence of these applications calls for a theoretical foundation for machine learning that accounts for such social and strategic interactions. This thesis presents some of the author's work towards developing such a foundation for machine learning by developing tools in the theory of machine learning and algorithmic economics. In short, the central theme of this thesis is

. . . to develop theoretical foundations for machine learning by the people, for the people.

1.1 Background

Before discussing the contributions of this thesis in more depth, let us present a brief overview of some of the basic concepts and frameworks that we use or contribute to.

1.1.1 Stackelberg Games

One of the commonly used game theoretic models in practice is the *Stackelberg game* model. This theoretical model has been applied for the purpose of computing optimal policies governing people and organizations, such as to fight crime, secure the borders [256], protect the environment [116], and manage supply chains and shelf space allocation [159]. The basic insight behind many of these applications is that the interactions between a policy maker and people form a

game in which the policy maker (first player) commits to a policy and a person (second player) observes this policy and responds by taking actions that benefit him most. From the viewpoint of the policy maker, the goal is to find the optimal strategy—the strategy that maximizes the policy maker's payoff when people best-respond.

More formally, a *Stackelberg game* is a two-player game with sets $\mathcal{X}$ and $\mathcal{Y}$ denoting the set of pure strategies available for the leader (player 1) and follower (player 2), respectively, and the set of outcomes $\mathcal{X} \times \mathcal{Y}$. Players have values over the set of outcomes, with $u_1(x, y)$ and $u_2(x, y)$ denoting the leader's and the follower's value for the pair of strategies (x, y). Given a defender *mixed strategy*, that is, a distribution $P \in \Delta(\mathcal{X})$ over pure strategies of the leader, the follower responds by taking the strategy that maximizes its expected payoff, i.e., the *best response*

$$b(P) = \arg\max_{y \in \mathcal{Y}} \mathop{\mathbb{E}}_{x \sim P}[u_2(x, y)].$$

The goal of the leader is to commit to a mixed strategy that leads to the highest expected payoff when the follower best-responds. That is to compute

$$\arg\max_{P \in \Delta(\mathcal{X})} \mathop{\mathbb{E}}_{x \sim P}[u_1(x, b(P))].$$

1.1.2 Offline Learning

Offline learning is one of the most classical problems in the theory of machine learning. In this setting, the learner has to learn to classify instances that are generated by a fixed joint distribution over labeled instances.

A common model of offline learning is the *agnostic supervised learning* model. In this model, there is an instance space $\mathcal{X}$, a label set $\mathcal{Y} = \{-1, +1\}$, and a hypothesis class $\mathcal{H}$, such that for each $h \in \mathcal{H}$, $h : \mathcal{X} \to \mathcal{Y}$. There is an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The learner has access to a set S of training samples $(x_1, y_1), \ldots, (x_m, y_m)$ that are drawn i.i.d from $\mathcal{D}$. For any h, the *true error* and *empirical error* of h are respectively defined by

$$\operatorname*{err}_{\mathcal{D}}(h) = \Pr_{(x,y) \sim \mathcal{D}}[h(x) \neq y]$$

and

$$\operatorname*{err}_{S}(h) = \frac{1}{m} \sum_{i \in [m]} \mathbb{I}_{h(x_i) \neq y_i},$$

where $\mathbb{I}$ is the indicator functions, i.e., for a boolean predicate b, $\mathbb{I}_b = 1$ when b is satisfied, and 0 otherwise.

The goal of the learner is to find a hypothesis $h \in \mathcal{H}$ that minimizes the true error. However, since the learner does not know $\mathcal{D}$, he has to choose a hypothesis by only considering the training sample set S. A classical example of such an algorithm is *Empirical Risk Minimization (ERM)* that returns the hypothesis with lowest empirical error. That is, it returns $h_S = \arg\min_{h \in \mathcal{H}} \operatorname{err}_S(h)$ with the hope that the true error of h_S is close to the optimal true error. A classical result from learning theory, called the *uniform convergence property*, states that when the training sample set is large enough, with high probability the true error and empirical error of

any hypothesis, and as a result those of h_S, are close to each other [11]. More formally, with probability $1-\delta$, for all $h \in \mathcal{H}$,

$$\left|\operatorname*{err}_{\mathcal{D}}(h) - \operatorname*{err}_{S}(h)\right| \leq O\left(\sqrt{\frac{\mathrm{VCdim}(\mathcal{H}) + \ln(1/\delta)}{m}}\right),$$

where $\mathrm{VCDim}(\mathcal{H})$ is the VC dimension [263] of $\mathcal{H}$. As a direct consequence of this result, we have that with probability $1-\delta$,

$$\left|\operatorname*{err}_{\mathcal{D}}(h_S) - \operatorname*{arg\,min}_{h\in\mathcal{H}} \operatorname*{err}_{\mathcal{D}}(h)\right| \leq O\left(\sqrt{\frac{\mathrm{VCdim}(\mathcal{H}) + \ln(1/\delta)}{m}}\right).$$

Another common model of offline learning is the *realizable Probably Approximately Correct (PAC)* model. In this setting, there is a hypothesis $h^* \in \mathcal{H}$ that perfectly labels the instances, i.e., $\mathrm{err}_{\mathcal{D}}(h^*) = 0$. The goal is to use the training sample set S to find a classifier $h \in \mathcal{H}$ with error $\mathrm{err}_{\mathcal{D}}(h) \leq \epsilon$. Note that this is much weaker than the uniform convergence property, as it does not require the convergence of true and empirical errors for all hypotheses $h \in \mathcal{H}$; rather, it is sufficient for those classifiers that perfectly label the training data set to also have a small true error. This allows us to obtain a stronger convergence bound. It is known that with probability $1-\delta$, for all $h \in \mathcal{H}$ such that $\mathrm{err}_S(h) = 0$, we have

$$\operatorname*{err}_{\mathcal{D}}(h) \leq O\left(\frac{1}{m}\left(\mathrm{VCdim}(\mathcal{H}) \ln\left(\frac{m}{\mathrm{VCdim}(\mathcal{H})}\right) + \ln\left(\frac{1}{\delta}\right)\right)\right).$$

1.1.3 Online Learning

Many situations involve learning in an environment that keeps changing and evolving in uncertain ways. Therefore, there is a need to develop adaptive learning tools that are robust to changes in the environment. This is where the *online learning* framework comes into play; it allows one to create online adaptive algorithms with performance guarantees that hold even when the environment is changing rapidly and adversarially.

We consider the following online learning problem. On each round $t = 1, \ldots, T$, a learner chooses an action $x_t \in \mathcal{X}$ and an adversary chooses an action $y_t \in \mathcal{Y}$. There is a fixed reward function $f : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ that is known to the learner. We consider two variants of online learning. First, the *full information* variant where the learner observes action y_t before the next round and receives a payoff of $f(x_t, y_t)$. Second, the *partial information* variant where the learner only observes some partial information about y_t, e.g., the learner may only observe the payoff $f(x_t, y_t)$. In both cases, the goal of the learner is to obtain low expected regret with respect to the best action in hindsight, i.e., to minimize

$$\textsc{Regret} \coloneqq \mathbb{E}\left[\max_{x\in\mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right],$$

where the expectation is over the randomness of the learner. We desire algorithms, called *no-regret algorithms*, for which this regret is sublinear in the time horizon T, equivalently, the average regret $\to 0$ as $T \to \infty$.

Online Learning	Learner action $x_t \in \mathcal{X}$	Adversary action $y_t \in \mathcal{Y}$	Payoff f
Online Linear Opt.	Vector $\mathbf{x}_t \in \mathcal{K}$	Cost vector $\mathbf{c}_t \in [0,1]^d$	$-\mathbf{x}_t \cdot \mathbf{c}_t(x_t)$
Online Prediction	Hypothesis $h_t \in \mathcal{H}$	$(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$	$-\mathbb{I}_{h(x_t) \neq y_t}$
Online Stackelberg	Mixed strategy $P_t \in \Delta(\mathcal{X})$	Attacker type $\theta_t \in \Theta$	$\mathbb{E}_{x \sim P_t}\left[u_1(x, b_{\theta_t}(P_t))\right]$
Online Auctions	Auction $a_t \in \mathcal{A}$	Bid profile $\mathbf{v}_t \in [0,1]^n$	$\mathrm{rev}(a_t, \mathbf{v}_t)$

Table 1.1: Applications of online learning to linear optimiation, Stackelberg games, auctions, and prediction.

The study of online no-regret algorithms goes back to the seminal works of Hannan [146] and Blackwell [48, 49], who developed algorithms with regret $\mathrm{poly}(|\mathcal{X}|)o(T)$ for the full information setting. Subsequently, Littlestone and Warmuth [195], Freund and Schapire [124], and Vovk [266] improved this by introducing algorithms with $\sqrt{T \log(|\mathcal{X}|)}$ regret. When the set of actions available to the learner is structured, this bound can be improved to $\sqrt{T\,\mathrm{Ldim}(\mathcal{X})}$, where $\mathrm{Ldim}(\mathcal{X})$ refers to the Littlestone dimension of $\mathcal{X}$ [194]. It is well-known that $\mathrm{Ldim}(\mathcal{X}) \leq \log(|\mathcal{X}|)$ in general, but in many cases, including some infinitely large classes, the Littlestone dimension is much smaller than $\log(|\mathcal{X}|)$. For the partial information setting, Auer et al. [21] introduced an algorithm with $\sqrt{T|\mathcal{X}|\log(|\mathcal{X}|)}$ regret. This was later improved by Bubeck et al. [66] who introduced an algorithm with regret $\sqrt{T|\mathcal{X}|}$.

Now, years after its inception in the works of Blackwell [48, 49], Hannan [146], the pressing need for robust learning algorithms in a wide range of problems is the driving force behind much of the recent progress in this area: with applications ranging from the more classical online linear optimization and online prediction problems, to the more modern applications of this domain to game theory and economics [27, 31, 52, 70, 73, 239], including our work on online Stackelberg games [37] and online auctions [113]. Here, we briefly describe a few examples of the domains in which online learning has played a major role.

Online Linear Optimization Given a convex region $\mathcal{K} \subset \mathbb{R}^d$, at every round the learner chooses a vector $\mathbf{x}_t \in \mathcal{K}$ and the adversary chooses a cost vector $\mathbf{c}_t$. The learner's payoff is $-\mathbf{x}_t \cdot \mathbf{c}_t$. (See chapter 6 for more details.)

Online Prediction Given an instance space $\mathcal{X}$ and a hypothesis class $\mathcal{H}$, such that every $h \in \mathcal{H}$ is a hypothesis $h : \mathcal{X} \to \{-1, +1\}$, at every round the learner chooses one hypothesis $h_t \in \mathcal{H}$. The adversary reveals an instance $x_t \in \mathcal{X}$ with label $y_t \in \{-1, +1\}$. The learner receives utility 0 if its prediction $h_t(x_t)$ matches y_t, and -1 otherwise, i.e., the learner's payoff is $-\mathbb{I}_{h(x_t) \neq y_t}$. (See chapter 7 for more details.)

Online Stackelberg Security Games In Stackelberg Security games, sometimes a defender has to face multiple types of attackers over time, each of whom have different preferences over targets. Given a set $\mathcal{N}$ of targets, defender strategy set $\mathcal{X}$, defender utility function u_1, a set of attacker types Θ where each $\theta \in \Theta$ represents an attacker with utility function u_θ. At every round, the defender chooses one mixed strategy $P_t \in \Delta(\mathcal{X})$, the Nature reveals a type of attacker $\theta_t \in \Theta$ with a corresponding best response function

$b_{\theta_t}(P_t) = \arg\max_{i \in \mathcal{N}} \mathbb{E}_{x \sim P_t}[u_{\theta_t}(x, i)]$. The defender's payoff is $\mathbb{E}_{x \sim P_t}[u_1(x, b_{\theta_t}(P_t))]$. (See chapter 4 for more details.)

Online Auctions In design of auctions, auctioneers often set the parameters of an auction, e.g., reserve prices, based on the preferences of bidders that they typically face. Since people's preferences change over time, auctioneers need to adaptively tailor auction parameters based on these changes. Given a set of auctions $\mathcal{A}$, where each auction $a \in \mathcal{A}$ takes valuations of n bidders, at every round the auctioneer chooses an auction $a_t \in \mathcal{A}$. The adversary chooses a valuation profile $\mathbf{v}$ of n bidders. The auctioneer receives revenue $\mathrm{rev}(a_t, \mathbf{v}_t)$. (See chapter 5 for more details.)

1.2 Overview of Thesis Contributions and Structure

The thesis presents a selection of the author's work on the theoretical aspects of machine learning and algorithmic economics, that contributes to a theory of machine learning and algorithmic economic that accounts for learning in presence of social and strategic behavior. This thesis is organized in three parts: learning about people, from people, and by people, respectively.

Learning About People

In their interactions with deployed systems and mechanisms, people reveal important information about their social and strategic behavior, their likes and dislikes, and, broadly speaking, their decision making process. Computational thinking has profoundly affected how we view these day-to-day interactions. A prime example of this is the use of Stackelberg games for the purpose of modeling and understanding interactions between strategic entities, such as people, organizations, and even algorithms.

A common application of the Stackelberg game model is to the physical security domain. In these *Stackelberg Security games*, the defender commits to a randomized deployment of his resources for protecting a set of potential targets, that is, $\mathcal{X}$ represents all possible deterministic deployments of resources to targets. The attacker responds by attacking a target, that is, $\mathcal{Y}$ represents the set of all targets, hereafter denoted by $\mathcal{N}$. When a target is attacked, both defender and attacker receive payoffs that depend on whether or not the target was protected in the defender's deployment. The attacker, having had surveillance of the defender's randomized deployment, attacks the target that maximizes his expected payoff. The goal is to compute an optimal defender strategy—one that would maximize the defender's payoff under the attacker's best response.

While the foregoing model is elegant, implementing it requires a lot of information in the form of an accurate model of people's strategic preferences, $u_1(x, y)$ and $u_2(x, y)$. Since any optimal policy that one computes can be at most as accurate as the model of behavior it receives as input, it is essential to create an accurate model of people's behavior. In Chapters 2, 3, and 4, we show how, through his interactions with the attacker, the defender can learn a sufficiently accurate model of attacker behavior to guide him in finding a near optimal defender strategy.

Chapter 2: Learning in Stackelberg Security Games In this chapter, we consider a Stackelberg Security game for which the payoffs of the attacker are unknown. We consider a learning-theoretic approach to dealing with uncertain attacker payoffs. We show that the defender can learn a near optimal strategy against an attacker by adaptively and iteratively committing to different strategies and observing the attacker's sequence of responses. In other words, we consider a setting where the attacker utility function u_2 is unknown, but one can still compute $b(P)$ for a given mixed strategy P by using it and observing the attacker's response. We call each round of committing to a strategy and observing the attacker's response a *query*. The algorithm we design uses $\text{poly}\left(|\mathcal{N}| \log\left(\frac{1}{\epsilon\delta}\right)\right)$ queries and returns a mixed strategy $P' \in \Delta(\mathcal{X})$ such that with probability $1-\delta$,

$$\max_{P \in \Delta(\mathcal{X})} \mathop{\mathbb{E}}_{x \sim P}[u_1(x, b(P))] - \mathop{\mathbb{E}}_{x \sim P'}[u_1(x, b(P'))] \leq \epsilon.$$

Our approach provides a practical method for calibrating the defender's strategy using a relatively short training period, and is especially appropriate for routine security tasks, e.g., ticket checks on public transportation.

One highlight of our result is that the number of queries our approach uses is polynomial in the number of targets, $|\mathcal{N}|$, independently of the total number of pure strategies that are available to the defender, $|\mathcal{X}|$. This is crucial because in most settings the number of pure strategies of the defender, i.e., the number of possible deterministic deployments of the defender's resources to targets, is exponential in the number of targets. In comparison, existing works in learning in Stackelberg games [190] had introduced algorithms that use $\text{poly}(|\mathcal{X}|)$ number of queries, making them unsuitable for Stackelberg Security games and other domains with large strategy spaces.

Chapter 3: Learning about a Boundedly Rational Attacker in Stackelberg Games In this chapter, we consider the problem of learning an optimal defender strategy in a Stackelberg Security game where the attacker may not be fully rational. Study of rationality in decision making has a long history in economics and social sciences [67, 166, 207, 240]. In many applications, it has been observed that strategic entities are not fully rational—at times they may take actions that are sub-optimal. In Stackelberg Security games, while deployments against sophisticated adversaries have often assumed that the adversary is a perfectly rational player who maximizes his expected value, it has been shown that such an assumption is not ideal for addressing less sophisticated human adversaries [214].

In the application of Stackelberg Security games to wildlife preservation, where the goal is to protect endangered animals from poachers, it has been observed that poacher behavior is best described by a model of rationality called *Subjective Utility Quantal Response* [214]. In this model, rather than attacking the target with the highest expected payoff in response to the mixed strategy P, the attacker may attack any target $i \in \mathcal{N}$ with probability

$$D^P(i) \propto \exp\left(\mathop{\mathbb{E}}_{x \sim P}[u_2(x, i)]\right).$$

For this model, we show that one can learn an accurate model of attacker behavior, hence, and accurate optimal defender strategy, by observing how the attacker responds to only three defender

strategies over a long period of time. More formally, we show that any three sufficiently different strategies and $m = \text{poly}\left(|\mathcal{N}|\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries each are sufficient to learn a defender strategy that is (additively) ϵ-close to the optimal defender strategy, with probability $1-\delta$.

One highlight of this result is that it can use observations from *any three historically used* defender strategies; in contrast, the algorithm introduced in Chapter 2 has to adaptively design new strategies to query. This is especially appropriate for applications where even a short calibration and training period may be undesirable, but large amount of historical records is available to the defender. For example, when learning optimal patrolling policies for the purpose of limiting and reducing poaching activities, using sub-optimal policies during the training period may lead to loss of animals that are already close to extinction. On the other hand, there are historical records of implemented policies and observed poaching activities over many years. Our approach shows how these historical records can be used to learn the optimal patrolling policy with no need for further calibration.

Online Learning—Dealing with Changes in People's Behavior

As discussed, Chapters 2 and 3 focus on learning Stackelberg optimal strategies against one type of attacker. That is, they assume that the attacker's preferences remain the same during the learning process. In many cases, however, people's preferences develop over time and our mechanisms will eventually encounter types of behavior that they were not designed for. Therefore, there is a need to develop adaptive learning tools that are robust to changes in the environment. This is where the *online learning* framework comes into play; it allows one to create online adaptive algorithms with performance guarantees that hold even when the environment is changing rapidly and adversarially.

Chapter 4: Online Learning in Multi-attacker Stackelberg Games In this chapter, we consider the information theoretic aspects of online learning in multi-attacker Stackelberg security games. The methods discussed in Chapters 2 and 3 are designed to use repeated interactions with a single attacker to learn the missing payoff information and compute a near optimal Stackelberg strategy against that attacker. However, sometimes a defender has to face multiple types of attackers over time, each of whom have different preferences over targets. In this chapter, we use the online learning framework to deal with the challenge of not knowing what type of an attacker one may face at any time.

We provide two algorithmic results that apply to two different models of feedback. In the *full information* model, the defender plays a mixed strategy and observes the type of attacker that responds. This means that the algorithm can infer the attacker's best response to *any* mixed strategy, not just the one that was played. We show that, though the space of mixed strategies of the defender is continuous and there are an infinite number of choices available to a defender at every round, the defender can limit its choices to an appropriately designed set $\mathcal{E}$ of mixed strategies of size $|\mathcal{E}| = \exp(nk)$ without incurring any additional regret. As mentioned earlier, using the classical no-regret algorithms in the full information setting, we immediately get a no-regret algorithm with regret $O(\text{poly}(nk)\sqrt{T})$.

In the second model—the *partial information* model—the defender only observes which target was attacked at each round. An additional challenge here is that classical no-regret

algorithms in the partial information setting have a regret that is polynomial in the size of the learner's action set. Therefore, simply limiting the defender's choices to the set of mixed strategies $\mathcal{E}$ is not sufficient for obtaining a regret bound that is polynomial in n and k. Here our main technical result is to design a no-regret algorithm in the partial information model whose regret is bounded by $O(\text{poly}(nk)T^{2/3})$.

For both results we assume that the attackers are selected (adversarially) from a set of k *known* types. It is natural to ask whether no-regret algorithms exist when there are no restrictions on the types of attackers. We answer this question in the negative, thereby justifying the dependence of our bounds on k.

Chapter 5: Oracle-Efficient Online Learning and Auction Design In this chapter, we consider the computational aspect of online learning. We consider the problem of online learning with full-information both for the general learner payoff and the payoff structure in economic mechanisms. As discussed, there are general purpose online learning algorithms that achieve a regret bound of $O\left(\sqrt{T\log(|\mathcal{X}|)}\right)$ in the full information setting. However, these information-theoretically optimal learning algorithms require a runtime of $\Omega(|\mathcal{X}|)$. This makes them unsuitable for settings where the action space is exponential in the natural representation of the problem, such as online Stackelberg games and online auctions. In this chapter, we design computationally efficient no-regret algorithms for problems that satisfy a structural property that are shared by many economic mechanisms.

Our goal is not achievable without some assumptions on the problem structure. Since an online optimization problem is at least as hard as the corresponding *offline optimization problem* [71, 94], a minimal assumption is the existence of an algorithm that returns a near-optimal solution to the offline problem. We call such an offline optimization algorithm an *oracle*, and an efficient algorithm that uses these oracles as a blackbox, an *oracle-efficient* algorithm.

Indeed, much of the effort in the field of algorithm design has been dedicated to the problem of offline optimization. Powerful tools, such as LPs and SDPs, have been developed to solve such problems quickly and without enumerating all possible solutions. Even when theoretical guarantees are not achievable, in some cases there are highly optimized specialized tools that can solve offline optimization problems fast in practice. *Oracle-efficient* online algorithms are algorithms that directly tap into these existing offline optimization algorithms. In addition to being of theoretical interest, existence of oracle-efficient algorithms sends a clear practical message: *offline optimization tools that are already deployed in practice can be directly used to robustly solve optimization problems in changing environments.*

Oracle-efficient algorithms have been introduced for some restrictive action spaces, such as linear and submodular functions [27, 155, 167, 169], while their existence in general problem spaces has been refuted by Hazan and Koren [156]. So, there is a need to identify structural properties of a problem space that would allow one to design an oracle-efficient algorithm. In this chapter, we show that when the problem space satisfies a structural property it admits an oracle-efficient online learning algorithm. Here, we describe a weaker form of this result and defer the complete description of this structural property to Chapter 5: if there exist N adversary actions $y^{(1)}, \ldots, y^{(N)} \in \mathcal{Y}$ such that any pair of learner's actions $x, x' \in \mathcal{X}$ receive sufficiently different rewards for at least one $y^{(i)}$, then our algorithm has regret $O(N\sqrt{T}/\delta)$ and runs in time

$\text{poly}(N, T)$ where δ is the smallest difference between distinct rewards on any one of the N actions.

The second contribution of this chapter is to show that many economic mechanisms, including online auctions design, demonstrate the above property. Therefore, showing that our algorithm can be used to achieve no-regret online learning in a large class of economic mechanisms. This includes online optimization of VCG auctions with bidder-specific reserves, envy-free item pricing, and level auctions.

Stable versus Adversarial Environments: A Middle Ground

The need for robust learning algorithms has led to the creation of online learning algorithms with performance guarantees that hold even when the environment that the learner performs in changes adversarially. Having been designed to perform well in adversarial environments, however, many online learning algorithms have learning guarantees that are significantly worst than those in stable environments. Thus, a natural question is whether the full power of online learning algorithms is necessary in day-to-day applications where the changes in the environment may be undesirable but not necessarily adversarial. For example, this may be the case when there are uncertainties in an environment, such as measurement inaccuracies, that hinders the adversary's choice, or, when the learner has additional information regarding how the environment is evolving. Is it possible to obtain algorithms with improved learning guarantees in environments that are not fully adversarial? This is the question we answer in the Chapters 6 and 7.

Chapter 6: Online Learning with Side Information In this chapter, we study a variant of online linear optimization where the player receives a hint about the cost function at the beginning of each round.

Online linear optimization, as described in Table 1.1 is a canonical problem in online learning. Many online algorithms exist that are designed to have a regret of $O(\sqrt{T})$ in the worst-case which is known to be information theoretically optimal. While this worst-case perspective on online linear optimization has lead to elegant algorithms and deep connections to other fields, such as boosting [124] and game theory [12, 53], it can be overly pessimistic. In particular, it does not account for the fact that the player may have side-information that allows him to anticipate the upcoming cost functions and evade the $\Omega(T)$ regret lower bound. In this chapter, we go beyond this worst case analysis and consider online linear optimization when additional information in the form of a function that is correlated with the cost that is presented to the player.

More formally, we consider the online linear optimization setup, described in Table 1.1, in which at every round the learner chooses a vector $\mathbf{x}_t \in \mathcal{K}$ and the adversary chooses a cost function $\mathbf{c}_t$. We further assume that the player receives a *hint* before choosing the action on each round. The hint in our setting is a vector that is guaranteed to be weakly correlated with the cost functions, i.e., the player receives $\mathbf{v}_t \in \mathbb{R}^d$ such that $\mathbf{v}_t \cdot \mathbf{c}_t \geq \alpha \|\mathbf{c}_t\|_2$ for some small but positive α. For example, when the cost function does not change rapidly from one round to the next, $\mathbf{c}_{t-1}$ acts as a hint for round t. Other times, the learner can take a small sample from the cost function, for example, see one of its (non-zero) coordinate.

We show that the player can benefit from such a hint if the set of feasible actions, $\mathcal{K}$, is sufficiently round. Specifically, if the set is strongly convex, the hint can be used to guarantee a regret of $O(\log(T))$, and if the set is q-uniformly convex for $q \in (2, 3)$, the hint can be used to guarantee a regret of $o(\sqrt{T})$. In contrast, we establish $\Omega(\sqrt{T})$ lower bounds on regret when the set of feasible actions is a polyhedron.

Chapter 7: Smoothed Online Learning In this chapter, we consider a middle ground between offline and online learning using the framework of smoothed analysis. As discussed in Section 1.1, offline and online learnability are characterized by two notions of complexity of the hypothesis space: the VC dimension [264] and the Littlestone dimension [193], respectively. In many hypothesis classes, however, there is a large gap between these two notions of complexity, and as a result, there is a gap in our ability to learn in the offline and online setting. For example, it is well-known that the class of 1-dimensional threshold functions has a VC dimension of 1 and can be learned in the offline i.i.d. setting with convergence rate (equivalently, regret) of $O\left(\sqrt{T}\right)$, but the Littlestone dimension of this class is unbounded so learning in the online adversarial setting is impossible. In this chapter, we use the *smoothed analysis* framework of Spielman and Teng [252] as a middle ground between online and offline learnability that leads to fundamentally stronger learnability results, like those achievable in the offline setting, but is still robust to the presence of an adversary.

The idea behind the smoothed analysis framework is that the adversary first chooses an arbitrary (worst-case) input, which is then perturbed slightly by nature. Equivalently, an adversary is forced to choose an input distribution that is not overly concentrated, and the input is then drawn from the adversary's chosen distribution. In addition to being a theoretically interesting middle ground between online and offline learning, there is also a plausible narrative about why "real-world" environments are captured by this framework: even in a world that is out to get us, there are inevitable inaccuracies such as measurement error and uncertainties that *smooths* the environment.

More formally, we consider the standard online prediction setup described in Table 1.1, with the exception that at every round t the adversary chooses an arbitrary distribution $\mathcal{D}_t$ over $\mathcal{X} \times \{-1, +1\}$ with a density function over $\mathcal{X}$ that is pointwise at most $1/\sigma$ times that of the uniform distribution. Then, $(x_t, y_t) \sim \mathcal{D}_t$ is presented to the learner. We consider a *non-adaptive* adversary that specifies $\mathcal{D}_1, \ldots, \mathcal{D}_T$ in advance.

We show that there is an algorithm with expected regret of $O\left(\sqrt{\mathrm{VCdim}(\mathcal{H})\, T \ln(T/\sigma)}\right)$ against any non-adaptive σ-smooth adversary. This gives us an algorithm with guarantees that smoothly transition between a worst-case adversary (when $\sigma \to 0$) to a uniformly random adversary (when $\sigma = 1$) where the offline learning guarantees a regret of $O\left(\sqrt{\mathrm{VCdim}(\mathcal{H})\, T}\right)$. In this regard, our work highlights win-win scenario by introducing algorithms that are robust to "realistic" adversarial changes in the environment with regret bounds that are almost as good as those in the fully stochastic (offline) setting.

Part II: Learning from People

Over the last decade, research in machine learning and AI has seen tremendous growth, partly due to the ease with which we can collect and annotate massive amounts of data across various domains. This rate of data annotation has been facilitated in part by crowdsourcing tools, such as Amazon Mechanical Turk, that employ people across the world to perform small tasks that require human intelligence.

Human participation in data annotation has brought a number of challenges to the forefront of machine learning research, one of the greatest of which is how to deal with human limitations, such as lack of expertise and commitment. Lack of expertise in participants often leads to data sets that are highly noisy [164, 179, 267]. Moreover, standard techniques for improving the quality of these data sets put a heavy burden on individuals who may have limited time and interest in participation. Therefore, the learning environments that involve the crowd give rise to a multitude of design choices that do not appear in traditional learning environments. These include: what challenges does the high amount of noise typically found in curated data sets pose to the learning algorithms? How does the goal of learning from the crowd differs from the goal of annotating data by the crowd? How do learning and labeling processes interplay?

The standard approach to learning from the crowd has been to view the process of acquiring labeled data through crowdsourcing and the process of learning a classifier in isolation. Indeed, most works in the crowdsourcing domain focus solely on collecting high quality data without considering the nature of the learning task that is to be performed on that data. Unfortunately, when learning and generalization from data is considered, *high quality data does not necessarily translate to a highly accurate learned model.* That is, two data sets with the same noise rate can lead to learned hypotheses that are significantly different in their quality. This is due to statistical and computational challenges involved in learning from a noisy data set. Below, we demonstrate some of these challenges.

From the computational perspective, our ability to efficiently learn from a noisy data set depends, to a large degree, on the type of noise we face. On one extreme, there has been significant work on the difficult *adversarial noise models*, where an adversary can choose some $\eta < \frac{1}{2}$ fraction of the data points and "deterministically" corrupt their labels. This is a particularly difficult noise model with strong negative results. For example, it is known that an adversary can take a data set that is perfectly labeled by a halfspace and corrupt just 1% of the data in such a way that finding a halfspace with 51% accuracy is NP-Hard [140]. On the other extreme is the much simpler *random classification noise* [176] model where the label of each instance is flipped with probability exactly $\eta < \frac{1}{2}$. When considering learning halfspaces in the presence of random classification noise, it is well-known that one can learn a halfspace that is arbitrarily close to the optimal one in polynomial time [54].

From the information theoretic perspective, our ability to learn an accurate classifier using small data sets depends on the type of noise we face. Take as an example Figure 1.2 where the data can be perfectly labeled by the red halfspace. In presence of adversarial noise (demonstrated in the middle) even with infinitely many samples (or knowing the full distribution of corrupted instances) the learner cannot distinguish the true halfspace—whether the blue or the red halfspace was the original classifier.[1] In the presence of random classification noise, however, the learner

[1] In presence of adversarial noise, rather than recovering the true halfspace—one that is accurate on the non-noisy

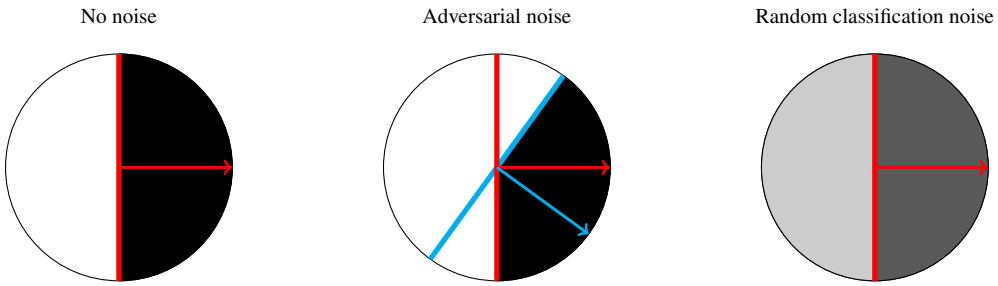


Figure 1.2: The figure on the left demonstrates instances labeled black and white that are perfectly classified by a halfspace. The figure in the middle demonstrates adversarial noise, where the adversary deterministically flips the labels of 10% of the data (the shaded region). The figure on the right demonstrates the random classification noise where the labels of all points (the shaded area) are flipped with probability 10%.

can find a d-dimensional halfspace that is ϵ-close in angle to the correct one using a data set of size $O\left(\frac{d}{\epsilon}\right)$ [65].

When considering crowdsourced learning models, perhaps neither the adversarial noise model nor the random classification noise model present a convincing model of the crowd's shortcomings. In this part of the thesis, we look at more realistic noise models that may arise from human participation in a classification task. We consider both scenarios where a noisy crowdsourced data set is given to us and our goal is to learn a classifier and scenarios where we can design the data annotation protocol as well as the learning algorithm.

Chapter 8: Learning with Bounded Noise In this chapter, we consider a setting where we are given a noisy data set and our goal is to learn an accurate classifier. We consider the *bounded noise* model, also known as *Massart noise* [65] or *Malicious Misclassification noise* [231, 251]. Bounded noise can be thought of as a generalization of the random classification noise model where the label of each example x is flipped independently with probability $\eta(x) < \frac{1}{2}$. That is, the adversary has control over choosing a different noise rate $\eta(x) \leq \eta$ for every example x with only the constraint that $\eta(x) \leq \eta$.

In addition to being of theoretical interest as a middle ground between the adversarial and random classification noise models, there is also a plausible narrative about why the noise in crowdsourced data sets is captured by this model: Consider the PAC learning framework, where there is a hypothesis $h^* \in \mathcal{H}$ that perfectly labels the data. Consider a large crowd of labelers, $1-\eta$ fraction of whom know the target function h^* and η fraction may make mistakes in arbitrary ways. Note there may be easy instances, i.e., many of the imperfect labelers label them correctly; and more difficult instances, i.e., only the perfect labelers know their correct label. So, any instance x that is labeled by a randomly chosen person from this crowd receives an incorrect label with probability $\eta(x) \leq \eta$, where the instance-dependent noise rate $\eta(x)$ captures the varying degree of difficulty of an instance.

distribution—the learner instead can learn a halfspace whose error rate on the noisy distribution is close to the error rate of the optimal classifier.

From the information theoretic point of view, similar to the case of random classification noise, it is well known that the learner can learn a classifier that is ϵ-close to being optimal using $O\left(\frac{\mathrm{VCdim}}{\epsilon}\right)$ samples [65]. From the computational perspective, due to its highly asymmetric nature, no computationally efficient learning algorithms had been known (except for classes with constant VCdim) until the work we present in this chapter. In this chapter, we provide the first computationally efficient algorithm in this noise model. In particular, we consider a setting where the marginal distribution over instances $\mathcal{X}$ constitutes an isotropic log-concave distribution in $\mathbb{R}^d$ and $\mathcal{H}$ is a class of d-dimensional halfspaces. We give an algorithm that for any ϵ and d takes $\mathrm{poly}(\frac{1}{\epsilon}, d)$ samples and in time $\mathrm{poly}(\frac{1}{\epsilon}, d)$ returns a classifier $h \in \mathcal{H}$ with excess error ϵ compared to h^*.

Chapter 9: Efficient Learning from the Crowd In this chapter, we focus on learning settings where we can design the learning algorithm as well as the data annotation protocol. We explore the crowdsourced setting behind the Bounded noise model of Chapter 8, where we have access to a large pool of labelers, an α (equivalently, $1-\eta$) fraction of whom are perfect labelers who never make a mistake, and others may make mistakes in arbitrary ways. As opposed to Chapter 8, where the data was collected by a third party and there were no direct interactions with the crowd, we consider a setting where we can actively query a labeler sampled form the crowd on specific instances drawn from the underlying distribution. This allows us to learn and acquire labels in tandem.

Our goal is to design learning algorithms that efficiently learn highly accurate classifiers using only a few queries. We compare the computational and statistical aspects of our algorithms to their PAC counterparts in the realizable setting as discussed in Section 1.1. As we know, $m_\epsilon^{PAC} = O\left(\frac{d}{\epsilon}\ln\left(\frac{1}{\epsilon}\right)\right)$ correctly labeled samples are sufficient to learn a classifier with ϵ-error in the realizable setting when $d = \mathrm{VCdim}(\mathcal{H})$. We look for algorithms that ask not many more queries than m_ϵ^{PAC}. Furthermore, we look for algorithms that can be performed efficiently if the corresponding learning task can be performed efficiently in the realizable setting. That is, we have access to an *oracle* that for any labeled data set returns a classifier from $\mathcal{H}$ that is consistent with that data set if one exists. Note that such algorithms are readily available for many hypothesis classes, such as using Perceptron for halfspaces.

Our results show that there is an *oracle-efficient* learning algorithm that can learn an ϵ-accurate classifier from the noisy crowd using $\frac{1}{\alpha}m_\epsilon^{PAC}$ queries [2] In other words, if $\mathcal{H}$ can be efficiently learned in the realizable PAC model, then it can be efficiently learned in the noisy crowdsourcing model with $O(\frac{1}{\alpha})$ queries per example. Additionally, each labeler is asked to label only $O(\frac{1}{\alpha})$ examples.

The above result highlights the importance of performing data annotation and learning in tandem. Recall that the bounded noise model of Chapter 8 corresponds to asking exactly 1 query per example from the crowd, for $\alpha > \frac{1}{2}$. As seen from the literature on learning theory and our work in Chapter 8, obtaining computationally efficient learning algorithms that are robust to Bounded noise, even for very simple hypothesis classes, has been a long standing open problem with positive results that only holds under restrictive assumptions, e.g., log-concave distributions with halfspaces [24, 25]. In comparison, the results in this chapter show that by asking a constant

[2] When $\alpha < \frac{1}{2}$, we need access to an expert to label $\frac{1}{\alpha}$ points correctly.

number of queries per example, instead of just one, we can devise efficient and general-purpose learning algorithms that work across the board.

Part III: Learning by People

With the wide application of machine learning methods to many aspects of day-to-day life, many simultaneous learning processes may be analyzing the same or related concepts at any given moment. This brings about a natural question: How do interactions between learners affect the learning process?

Chapter 10: Collaborative Learning In this chapter, we consider a setting where learners who are interested in performing related, but not necessarily the same, learning tasks can collaborate and share information to make the learning process more efficient. It is self-evident that collaboration is beneficial for learning, but how beneficial? This is what we formalize in this chapter.

We consider a model of *collaborative PAC learning*, in which k players attempt to learn the same underlying concept. We then ask how much information is needed for all players to simultaneously succeed in learning desirable classifiers. Specifically, we focus on the classic *probably approximately correct (PAC)* setting of Valiant [262], where there is an unknown target function $h^* \in \mathcal{H}$. We consider k players with distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$ that are labeled according to h^*. Our goal is to learn h^* up to an error of ϵ on *each and every* player distribution while requiring only a small number of samples overall.

We compare the number of samples needed to accomplish all k learning tasks in the collaborative setting to the number of samples needed if no collaboration happens between the learners. Using the PAC learning sample bounds from Section 1.1, it is evident that the latter requires $O\left(k\epsilon^{-1}\mathrm{VCdim}(\mathcal{H})\ln\left(\frac{1}{\epsilon}\right)\right)$ samples. Our main technical result in this chapter is that when learners collaborate the total number of samples needed to accomplish all k tasks is $O\left(\log(k)\epsilon^{-1}\mathrm{VCdim}(\mathcal{H})\ln\left(\frac{1}{\epsilon}\right)\right)$. Furthermore, we show that $\Theta\left(\log(k)\epsilon^{-1}\mathrm{VCdim}(\mathcal{H})\ln\left(\frac{1}{\epsilon}\right)\right)$ samples are needed, even in the collaborative settings, for all learners to accomplish their learning task.

Part IV: Learning For People

Pressing practical and societal needs, such as security and organ transplant, have inspired the design of theoretical models. However, the uncertainties that arise from transitioning these models from theory to practice can quickly degrade the quality of solutions that seem effective in theory. This is where machine learning comes in; it robustly addresses these uncertainties by gathering additional information when needed or establishing that the existing models are resilient to the environment's uncertainty. As part of *learning about people* in Part I of this thesis, we discussed how machine learning can help us create better mechanisms for physical security. In this part, we consider another application of great societal importance: *kidney exchange*.

The best treatment for people who suffer from chronic kidney disease is transplanting a healthy kidney. Even for those patients that are fortunate enough to have a willing live donor,

usually a family member, direct transplantation is not always a viable option due to medical incompatibility issues. This is where *kidney exchange* comes in. In its simplest form, it allows two incompatible donor-patient pairs—such that the first patient is compatible with the second donor and the second patient is compatible with the first donor—to swap donors, so that each patient receives a compatible kidney. More generally, kidney exchange allows a cycle of donor-patient pairs (up to the size 3 in practice), where each patient is compatible with the donor of the next pair, to exchange kidneys.

In this part of the thesis, we consider the challenges that arise from deploying kidney exchange models in practice. In particular, we consider two sources of uncertainty: the mechanism's lack of information on the compatibility between different patient-donor pairs and the hospital's lack of information on whether or not collaborating with other hospitals would benefit their patients.

Chapter 11: A Near Optimal Kidney Exchange with a few Queries A successful transplant procedure relies mainly on two compatibility tests: First, a test that checks the blood type and tissue type of the donor and patient, and, second, a *crossmatch* test that examines the compatibility of a donor-patient pair by physically mixing their blood. Due to the significant cost of crossmatch tests, kidney exchange programs perform at most one crossmatch per patient. Consequently, most seemingly feasible matches based on blood type and tissue type fail at the crossmatch level. On average, this results in less than 10% of patients receiving a kidney transplant. On the other hand, a hypothetical procedure that performs all crossmatch tests—call it the *omniscient optimum*—is much more effective, but its costly implementation renders it impractical.

In this chapter, we provide a polynomial time algorithm that for any $\epsilon > 0$, proceeds in $O_\epsilon(1)$ rounds, at each round performing 1 crossmatch test per patient, for an overall constant $O_\epsilon(1)$ number of crossmatches per patient[3]. Our algorithm then recovers a $(1-\epsilon)$ fraction of the omniscient optimum for 2-way exchanges, and $\frac{4}{9}(1-\epsilon)$ fraction of the omniscient optimum for 3-way exchanges. Our results send a clear conceptual message: *with a mild change at the policy level to the number of crossmatch tests that are performed—from one to a few—we can effectively get the full benefit of exhaustive testing at a fraction of the cost.*

This approach is related to the problem of membership query learning in learning theory[4], where a target function is queried at points of an algorithm's choosing in order to identify the function. In this case, however, the goal is not to fully identify the underlying true kidney exchange graph, but rather just enough to produce a near-optimal matching.

Chapter 12: Individually Rational Multi-Hospital Kidney Exchange Designing a near optimal kidney exchange mechanism that takes into account the incentives of the participants has become increasingly more important. In recent years, hospitals have enrolled patients into regional or even national kidney exchange programs. However, hospitals may choose not to participate. This is usually due to the fact that hospitals are uncertain whether they can match many more of their own patients on their own. Economists would say that the exchange may not be *individually rational* for the hospitals.

[3] $O_\epsilon(1)$ denotes a time complexity that is constant when ϵ is considered to be a constant. Importantly, $O_\epsilon(1)$ has no dependence on the size of the graph in this context

[4] Similar to those used in the Stackelberg security games in Chapter 2

This fear of uncertainty regarding how many of their patients can be matched by other hospitals has driven hospitals to take a worst case viewpoint. Indeed, creating a globally optimal kidney exchange mechanism is at odds with hospital's goal of matching as many as their own patients when considering the worst case compatibility between patients and the worst-case assignment of patients to hospitals. In this case, a globally optimal exchange may only match $\frac{2}{3}$ of the patients that a hospital could have matched locally and individually rational kidney exchange mechanisms match at most $\frac{1}{2}$ of the patients matched in the globally optimal exchange.

In this chapter, we offer a new perspective on this problem. Our key insight is that it suffices to assume that assignment of patients to hospitals is a random process independent of the compatibility of them with other patients. The rationale for this model is simple: there is no reason as to why donor-patient pairs with particular medical compatibility would belong to a particular hospital the probability of that happening depends primarily on the size of the hospital. Taking this view, we show that the uncertainty affecting the hospital's number of matched patients is indeed quite small. That is, with high probability over the assignment of patients to hospitals, every hospital receives *almost* as many matched patients in the global matching that it could have matched by itself. To complement these results, we also provide a mechanism that with complete certainty is fully individually rational and with high probability is close to being globally optimal.

While our approach to this problem is not directly in the framework of machine learning, we take use of statistical learning theory tools, such as convergence bounds for combinatorial functions, to limit the degree of uncertainty that a hospital faces when joining a multi-hospital kidney exchange.

1.3 Bibliographical Remarks

The research presented in this thesis is based on joint work with several co-authors, described below. This thesis only includes works for which this author was the, or one of the, primary contributors.

Chapter 2 is based on joint work with Avrim Blum and Ariel D. Procaccia [56]. Chapter 3 is based on joint work with Fei Fang, Thanh H. Nguyen, Ariel D. Procaccia, Arunesh Sinha, and Milind Tambe [142]. Chapter 4 is based on joint work with Maria-Florina Balcan, Avrim Blum and Ariel D. Procaccia [37]. Chapter 5 is based on joint work with Miroslav Dudik, Haipeng Luo, Robert E. Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan [113]. Chapter 6 is based on joint work with Ofer Dekel, Authur Flajolet, and Patrick Jaillet [101]. Chapter 7 is based on ongoing joint work with Tim Roughgarden. Chapter 8 is based on joint works with Pranjal Awasthi, Maria-Florina Balcan, Ruth Urner, and Hongyang Zhang [24, 25]. Chapter 9 is based on a joint work with Pranjal Awasthi, Avrim Blum, and Yishay Mansour [26]. Chapter 10 is based on a joint work with Avrim Blum, Ariel D. Procaccia, and Mingda Qiao [60]. Chapter 11 is based on a joint work with John P. Dickerson, Avrim Blum, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma [58]. Finally, Chapter 12 is based on a joint work with Avrim Blum, Ioannis Caragiannis, Ariel D. Procaccia, Eviatar B. Procaccia, and Rohit Vaish [59].

1.4 Excluded Research

In an effort to keep this dissertation succinct and coherent, a significant portion of this author's Ph.D. work has been excluded from this document. The excluded research includes:

- Work on topic modeling: Bridging the gap between co-training and topic modeling [51].
- Work on security domains: Monitoring Stealthy Diffusion, Dealing with attackers's lack of accurate observation [57], Computational aspect of Large Stackelberg Games.
- Online optimization of combinatorial objects: Voting rules [144] and Minmax Submodular functions [7].
- Work on motion planning using few queries [143].
- Work on clustering [38].

Part I

Learning about People

Chapter 2

Learning in Stackelberg Security Games

2.1 Introduction

In this chapter, we consider learning in Stackelberg Security games. In the past decade, Stackelberg games have been used by the U.S. Coast Guard, the Federal Air Marshal Service, the Los Angeles Airport Police, and other major security agencies for the purpose of finding good security deployments [256]. The idea behind the application of the Stackelberg game model to the physical security domain is simple: the interaction between the *defender* and a potential *attacker* can be modeled as a *Stackelberg game*, in which the defender commits to a (possibly randomized) deployment of his resources, and the attacker responds in a way that maximizes his own payoff. The algorithmic challenge is to compute an optimal defender strategy—one that would maximize the defender's payoff under the attacker's best response.

While the foregoing model is elegant, implementing it requires a significant amount of information. Perhaps the most troubling assumption is that we can determine the attacker's payoffs for different outcomes. In deployed applications, these payoffs are estimated using expert analysis and historical data—but an inaccurate estimate can lead to significant inefficiencies. The uncertainty about the attacker's payoffs can be encoded into the optimization problem itself, either through robust optimization techniques [224], or by representing payoffs as continuous distributions [178].

Letchford et al. [190] take a different, learning-theoretic approach to dealing with uncertain attacker payoffs. Studying Stackelberg games more broadly (which are played by two players, a *leader* and a *follower*), they show that the leader can *efficiently learn* the follower's payoffs by iteratively committing to different strategies, and observing the attacker's sequence of responses. In the context of security games, this approach may be questionable when the attacker is a terrorist, but it is a perfectly reasonable way to calibrate the defender's strategy for routine security operations when the attacker is, say, a smuggler. And the learning-theoretic approach has two major advantages over modifying the defender's optimization problem. First, the learning-theoretic approach requires no prior information. Second, the optimization-based approach deals with uncertainty by inevitably degrading the quality of the solution, as, intuitively, the algorithm has to simultaneously optimize against a range of possible attackers; this problem is circumvented by the learning-theoretic approach.

But let us revisit what we mean by "efficiently learn". The number of queries, i.e., observations of follower responses to leader strategies, required by the algorithm of Letchford et al. [190] is polynomial in the number of pure leader strategies. The main difficulty in applying their results to Stackelberg security games is that even in the simplest security game, the number of pure defender strategies is exponential in the representation of the game. For example, if each of the defender's resources can protect one of two potential targets, there is an exponential number of ways in which resources can be assigned to targets.

Our approach and results. We design an algorithm that learns an (additively) ϵ-optimal strategy for the defender with probability $1 - \delta$, by asking a number of queries that is polynomial in the representation of the security game, and logarithmic in $1/\epsilon$ and $1/\delta$. Our algorithm is completely different from that of Letchford et al. [190]. Its novel ingredients include:

- We work in the space of feasible coverage probability vectors, i.e., we directly reason about the probability that each potential target is protected under a randomized defender strategy. Denoting the number of targets by n, this is an n-dimensional space. In contrast, Letchford et al. [190] study the exponential-dimensional space of randomized defender strategies. We observe that, in the space of feasible coverage probability vectors, the region associated with a specific best response for the attacker (i.e., a specific target being attacked) is convex.

- To optimize within each of these convex regions, we leverage techniques—developed by Kalai and Vempala [170]—for optimizing a linear objective function in an unknown convex region using only membership queries. In our setting, it is straightforward to build a membership oracle, but it is quite nontrivial to satisfy a key assumption of the foregoing result: that the optimization process starts from an interior point of the convex region. We do this by constructing a hierarchy of nested convex regions, and using smaller regions to obtain interior points in larger regions.

- We develop a method for efficiently discovering new regions. In contrast, Letchford et al. [190] find regions (in the high-dimensional space of randomized defender strategies) by sampling uniformly at random; their approach is inefficient when some regions are small.

2.2 The Model

A Stackelberg security game is a two-player general-sum game between a *defender* (or the *leader*) and an *attacker* (or the *follower*). In this game, the defender commits to a randomized allocation of his security resources to defend potential targets. The attacker, in turn, observes this randomized allocation and attacks the target with the best expected payoff. The defender and the attacker receive payoffs that depend on the target that was attacked and whether or not it was defended. The defender's goal is to choose an allocation that leads to the best payoff.

More precisely, a security game is defined by a 5-tuple $(\mathcal{N}, \mathcal{D}, R, A, U)$:

- $\mathcal{N} = \{1, \ldots, n\}$ is a set of n *targets*.

- R is a set of *resources*.

- $\mathcal{D} \subseteq 2^{\mathcal{N}}$ is a collection of subsets of targets, each called a *schedule*, such that for every schedule $D \in \mathcal{D}$, targets in D can be simultaneously defended by one resource. It is natural to assume that if a resource is capable of covering schedule D, then it can also cover any subset of D. We call this property *closure under the subset operation*; it is also known as "subsets of schedules are schedules (SSAS)" [184].

- $A : R \to 2^{\mathcal{D}}$, called the *assignment function*, takes a resource as input and returns the set of all schedules that the resource is capable of defending. An allocation of resources is *valid* if every resource r is allocated to a schedule in $A(r)$.

- The *payoffs* of the players are given by functions $U_d(i, p_i)$ and $U_a(i, p_i)$, which return the expected payoffs of the defender and the attacker, respectively, when target i is attacked and it is covered with probability p_i. [1] We make two assumptions that are common to all works on security games. First, these utility functions are linear. Second, the attacker prefers it if the attacked target is not covered, and the defender prefers it if the attacked target is covered, i.e., $U_d(i, p_i)$ and $U_a(i, p_i)$ are respectively increasing and decreasing in p_i. We also assume w.l.o.g. that the utilities are normalized to have values in $[-1, 1]$. If the utility functions have coefficients that are rational with denominator at most a, then the game's (utility) representation length is $L = n \log n + n \log a$.

A *pure strategy* of the defender is a valid assignment of resources to schedules. The set of pure strategies is determined by $\mathcal{N}$, $\mathcal{D}$, R, and A. Let there be m pure strategies; we use the following $n \times m$, zero-one matrix M to represent the set of all pure strategies. Every row in M represents a target and every column represents a pure strategy. $M_{ix} = 1$ if and only if target i is covered using some resource in pure strategy x. A *mixed strategy* (hereinafter, called strategy) is a distribution over the pure strategies. To represent a strategy we use a $1 \times m$ vector $\mathbf{s}$, such that s_x is the probability with which the pure strategy x is played, and $\sum_{x=1}^{m} s_x = 1$.

Given a defender's strategy, the *coverage probability* of a target is the probability with which it is defended. Let $\mathbf{s}$ be a defender's strategy, then the coverage probability vector is $\mathbf{p}^\top = M\mathbf{s}^\top$, where p_i is coverage probability of target i. We call a probability vector *implementable* if there exists a strategy that imposes that coverage probability on the targets.

Let $\mathbf{p}^{\mathbf{s}}$ be the corresponding coverage probability vector of strategy $\mathbf{s}$. The attacker's *best response* to $\mathbf{s}$ is defined by $b(\mathbf{s}) = \arg\max_i U_a(i, p_i^{\mathbf{s}})$. Since the attacker's best-response is determined by the coverage probability vector irrespective of the strategy, we slightly abuse notation by using $b(\mathbf{p}^{\mathbf{s}})$ to denote the best-response, as well. We say that target i is "better" than i' for the defender if the highest payoff he receives when i is attacked is more than the highest payoff he receives when i' is attacked. We assume that if multiple targets are tied for the best-response, then ties are broken in favor of the "best" target.

The defender's *optimal strategy* is defined as the strategy with highest expected payoff for the defender, i.e. $\arg\max_{\mathbf{s}} U_d(b(\mathbf{s}), p^{\mathbf{s}}_{b(\mathbf{s})})$. An optimal strategy $\mathbf{p}$ is called *conservative* if no other

[1] Using the language introduced in Chapter 1, $U_d(i, p_i) = \mathbb{E}_{j \sim \mathbf{p}}[u_1(j, i)]$ and $U_a(i, p_i) = \mathbb{E}_{j \sim \mathbf{p}}[u_2(j, i)]$, where $j \sim \mathbf{p}$ denotes choosing element j with probability p_i.

optimal strategy has a strictly lower sum of coverage probabilities. For two coverage probability vectors we use $\mathbf{q} \preceq \mathbf{p}$ to denote that for all i, $q_i \leq p_i$.

2.3 Problem Formulation and Technical Approach

In this section, we give an overview of our approach for learning the defender's optimal strategy when U_a is not known. To do so, we first review how the optimal strategy is computed in the case where U_a is known.

Computing the defender's optimal strategy, even when $U_a(\cdot)$ is known, is NP-Hard [183]. In practice the optimal strategy is computed using two formulations: Mixed Integer programming [219] and Multiple Linear Programs [85]; the latter provides some insight for our approach. The Multiple LP approach creates a separate LP for every $i \in \mathcal{N}$. This LP, as shown below, solves for the optimal defender strategy under the restriction that the strategy is valid (second and third constraints) and the attacker best-responds by attacking i (first constraint). Among these solutions, the optimal strategy is the one where the defender has the highest payoff.

$$\begin{aligned}
\text{maximize} \quad & U_d(i, \sum_{x:M_{ix}=1} s_x) \\
\text{s.t.} \quad & \forall i' \neq i,\ U_a(i', \sum_{x:M_{i'x}=1} s_x) \leq U_a(i, \sum_{x:M_{ix}=1} s_x) \\
& \forall x,\ s_x \geq 0 \\
& \sum_{x=1}^{n} s_x = 1
\end{aligned}$$

We make two changes to the above LP in preparation for finding the optimal strategy in polynomially many queries, when U_a is unknown. First, notice that when U_a is unknown, we do not have an *explicit* definition of the first constraint. However, *implicitly* we can determine whether i has a better payoff than i' by observing the attacker's best-response to $\mathbf{s}$. Second, the above LP has exponentially many variables, one for each pure strategy. However, given the coverage probabilities, the attacker's actions are independent of the strategy that induces that coverage probability. So, we can restate the LP to use variables that represent the coverage probabilities and add a constraint that enforces the coverage probabilities to be implementable.

$$\begin{aligned}
\text{maximize} \quad & U_d(i, p_i) \\
\text{s.t.} \quad & i \text{ is attacked} \\
& \mathbf{p} \text{ is implementable}
\end{aligned} \tag{2.1}$$

This formulation requires optimizing a linear function over a region of the space of coverage probabilities, by using membership queries. We do so by examining some of the characteristics of the above formulation and then leveraging an algorithm introduced by Kalai and Vempala [170] that optimizes over a convex set, using only an initial point and a membership oracle. Here, we restate their result in a slightly different form.

Proposition 2.3.1 (Theorem 2.1 of Kalai and Vempala [170])**.** *For any convex set $H \subseteq \mathbb{R}^n$ that is contained in a ball of radius R, given a membership oracle, an initial point with margin r in H, and a linear function $\ell(\cdot)$, with probability $1-\delta$ we can find an ϵ-approximate optimal solution for ℓ in H, using $O(n^{4.5} \log \frac{nR^2}{r\epsilon\delta})$ queries to the oracle.*

2.4 Main Result

In this section, we design and analyze an algorithm that (ϵ, δ)-learns the defender's optimal strategy in a number of best-response queries that is polynomial in the number of targets and the representation, and logarithmic in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. Our main result is:

Theorem 2.4.1. *Consider a security game with n targets and representation length L, such that for every target, the set of implementable coverage probability vectors that induce an attack on that target, if non-empty, contains a ball of radius $1/2^L$. For any $\epsilon, \delta > 0$, with probability $1-\delta$, Algorithm 2.2 finds a defender strategy that is optimal up to an additive term of ϵ, using $O(n^{6.5}(\log \frac{n}{\epsilon\delta} + L))$ best-response queries to the attacker.*

The main assumption in Theorem 2.4.1 is that the set of implementable coverage probabilities for which a given target is attacked is either empty or contains a ball of radius $1/2^L$. This implies that if it is possible to make the attacker prefer a target, then it is possible to do so with a small margin. This assumption is very mild in nature and its variations have appeared in many well-known algorithms. For example, interior point methods for linear optimization require an initial feasible solution that is within the region of optimization with a small margin [133]. Letchford et al. [190] make a similar assumption, but their result depends linearly, instead of logarithmically, on the minimum volume of a region (because they use uniformly random sampling to discover regions).

To informally see why such an assumption is necessary, consider a security game with n targets, such that an attack on any target but target 1 is very harmful to the defender. The defender's goal is therefore to convince the attacker to attack target 1. The attacker, however, only attacks target 1 under a very specific coverage probability vector, i.e., the defender's randomized strategy has to be just so. In this case, the defender's optimal strategy is impossible to approximate.

The remainder of this section is devoted to proving Theorem 2.4.1. We divide our intermediate results into sections based on the aspect of the problem that they address.

2.4.1 Characteristics of the Optimization Region

One of the requirements of Proposition 2.3.1 is that the optimization region is convex. Let $\mathcal{P}$ denote the space of implementable probability vectors, and let $\mathcal{P}_i = \{\mathbf{p} : \mathbf{p} \text{ is implementable and } b(\mathbf{p}) = i\}$. The next lemma shows that $\mathcal{P}_i$ is indeed convex.

Lemma 2.4.2. *For all $i \in \mathcal{N}$, $\mathcal{P}_i$ is the intersection of a finitely many half-spaces.*

Proof. $\mathcal{P}_i$ is defined by the set of all $\mathbf{p} \in [0,1]^n$ such that there is $\mathbf{s}$ that satisfies the LP with the following constraints. There are m half-spaces of the form $s_x \geq 0$, 2 half-spaces $\sum_x s_x \leq 1$

and $\sum_x s_x \geq 1$, $2n$ half-spaces of the form $M\mathbf{s}^\top - \mathbf{p}^\top \leq 0$ and $M\mathbf{s}^\top - \mathbf{p}^\top \geq 0$, and $n-1$ half-spaces of the form $U_a(i, p_i) - U_a(i', p_{i'}) \geq 0$. Therefore, the set of $(\mathbf{s}, \mathbf{p}) \in \mathcal{R}^{m+n}$ such that $\mathbf{p}$ is implemented by strategy $\mathbf{s}$ and causes an attack on i is the intersection of $3n+m+1$ half-spaces. $\mathcal{P}_i$ is the reflection of this set on n dimensions; therefore, it is also the intersection of at most $3n+m+1$ half-spaces. □

Lemma 2.4.2, in particular, implies that $\mathcal{P}_i$ is convex. The Lemma's proof also suggests a method for finding the minimal half-space representation of $\mathcal{P}$. Indeed, the set $S = \{(\mathbf{s}, \mathbf{p}) \in \mathbb{R}^{m+n} : \text{Valid strategy } \mathbf{s} \text{ implements } \mathbf{p}\}$ is given by its half-space representation. Using the Double Description Method [125, 211], we can compute the vertex representation of S. Since, $\mathcal{P}$ is a linear transformation of S, its vertex representation is the transformation of the vertex representation of S. Using the Double Description Method again, we can find the minimal half-space representation of $\mathcal{P}$.

Next, we establish some properties of $\mathcal{P}$ and the half-spaces that define it.

Lemma 2.4.3. *Let $\mathbf{p} \in \mathcal{P}$. Then for any $\mathbf{0} \preceq \mathbf{q} \preceq \mathbf{p}$, $\mathbf{q} \in \mathcal{P}$.*

Proof. Let there be k targets i such that $p_i > q_i$. We prove this lemma by induction on k. For $k = 0$, the lemma trivially holds. Assume that for all $k < k_0$ the result holds. Let $k = k_0$. Let i be an arbitrary target for which $p_i > q_i$. Since the set of pure strategies is closed under subset operation there is a map, $\sigma(\cdot)$, such that for every pure strategy M_x (a column in matrix M) such that $M_{ix} = 1$, $M_{\sigma(x)}$ only differs from M_x in the i^{th} row (target). Let X and X' indicate these strategies i.e. $X = \{x : M_{ix} = 1\}$, $X' = \{\sigma(x) : M_{ix} = 1\}$.

Define $\mathbf{s}'$ as follows: For all $x \in X$, let $s'_x = s_x \cdot \frac{q_i}{p_i}$ and $s'_{\sigma(x)} = s_{\sigma(x)} + s_x \cdot \frac{p_i - q_i}{q_i}$, and for any $x \notin X \cup X'$, let $s'_x = s_x$. Consider $\mathbf{p}'$ that is induced by $\mathbf{s}'$: For i, $p'_i = \sum_{x:M_{ix}=1} s'_x = \sum_{x \in X} s_x \cdot \frac{q_i}{p_i} = q_i$. For all $i' \neq i$,

$$
\begin{aligned}
p'_{i'} &= \sum_{x:M_{i'x}=1} s'_x = \sum_{\substack{x:M_{i'x}=1 \\ \text{and } x \in X}} \left(s'_x + s'_{\sigma(x)}\right) + \sum_{\substack{x:M_{i'x}=1 \\ \text{and } x \notin I \cup I'}} s'_x \\
&= \sum_{\substack{x:M_{i'x}=1 \\ \text{and } x \in I}} \left(\frac{q_i}{p_i} s_x + s_{\sigma(x)} + \frac{p_i - q_i}{p_i} s_x\right) + \sum_{\substack{x:M_{i'x}=1 \\ \text{and } x \notin I \cup I'}} s_x \\
&= \sum_{\substack{x:M_{i'x}=1 \\ \text{and } x \in I}} \left(s_x + s_{\sigma(x)}\right) + \sum_{\substack{x:M_{i'x}=1 \\ \text{and } x \notin I \cup I'}} s_x = \sum_{x:M_{i'x}=1} s_x \\
&= p_{i'}
\end{aligned}
$$

We conclude that $\mathbf{p}'$ such that $p'_i = q_i$ and for all $i' \neq i$, $p'_{i'} = p_i$, is implementable. $\mathbf{p}'$ and $\mathbf{q}$ differ in only $k_0 - 1$ indices and for all j, $p'_j \geq q_j$, so using the induction hypothesis $\mathbf{q}$ is implementable. □

Lemma 2.4.4. *Let A be a set of a positive volume that is the intersection of finitely many half-spaces. Then the following two statements are equivalent.*

1. *For all $\mathbf{p} \in A$, $\mathbf{p} \succeq \boldsymbol{\epsilon}$. And for all $\boldsymbol{\epsilon} \preceq \mathbf{q} \preceq \mathbf{p}$, $\mathbf{q} \in A$.*

2. *A can be defined as the intersection of $\mathbf{e}_i \cdot \mathbf{p} \geq \epsilon$ for all i, and a set H of half-spaces, such that for any $\mathbf{h} \cdot \mathbf{p} \geq b$ in H, $\mathbf{h} \preceq \mathbf{0}$, and $b \leq -\epsilon$.*

Proof. (1 $\implies$ 2). Consider the minimal set of half-spaces that defines A. We know that this set is unique, and is the collection of facet-defining half-spaces. Since A has a positive volume, for all i, $(\mathbf{e}_i, \epsilon)$ is a facet, so it belongs to the set of half-spaces that define A. Take any half-space $(\mathbf{h}, b)$ in this collection that is not of the form $(\mathbf{e}_i, \epsilon)$. There is a point $\mathbf{p}$ on the the boundary of $(\mathbf{h}, b)$ that is not on the boundary of any other half-space (including $(\mathbf{e}_i, \epsilon)$), so $\mathbf{p} \succ \boldsymbol{\epsilon}$. For every i, define $\mathbf{p}^i$ such that $p^i_i = \epsilon$ and $p^i_j = p_j$ for all $j \neq i$. Then,

$$\mathbf{h} \cdot \mathbf{p}^i = \mathbf{h} \cdot \mathbf{p} + \mathbf{h} \cdot (\mathbf{p}^i_i - \mathbf{p}) = b - h_i(p_i - \epsilon).$$

Since $\mathbf{p}^i \in A$, $h_i \leq 0$. Since, $\boldsymbol{\epsilon} \in A$, $\mathbf{h} \cdot \boldsymbol{\epsilon} = -\|\mathbf{h}\|_1 \epsilon \geq b$, so $b \leq -\epsilon$.

(2 $\implies$ 1). For any $\mathbf{p} \in A$ and any $\boldsymbol{\epsilon} \preceq \mathbf{q} \preceq \mathbf{p}$, and any i, $\mathbf{e}_i \cdot \mathbf{q} = q_i \geq \epsilon$. For any $(h, b) \in H$ of the second form,

$$\mathbf{h} \cdot \mathbf{q} = \mathbf{h} \cdot \mathbf{p} + \mathbf{h} \cdot (\mathbf{q} - \mathbf{p}) \geq b,$$

where the last transition is by the fact that $\mathbf{h}$ and $(\mathbf{q} - \mathbf{p})$ are non-positive. So, $\mathbf{q} \in A$. □

Using Lemmas 2.4.3 and 2.4.4, we can refer to the set of half-spaces that define $\mathcal{P}$ by $\{(\mathbf{e}_i, 0) : \text{for all } i\} \cup H_{\mathcal{P}}$, where for all $(\mathbf{h}^*, b^*) \in H_{\mathcal{P}}$, $\mathbf{h}^* \preceq \mathbf{0}$, and $b^* \leq 0$.

2.4.2 Finding Initial Points

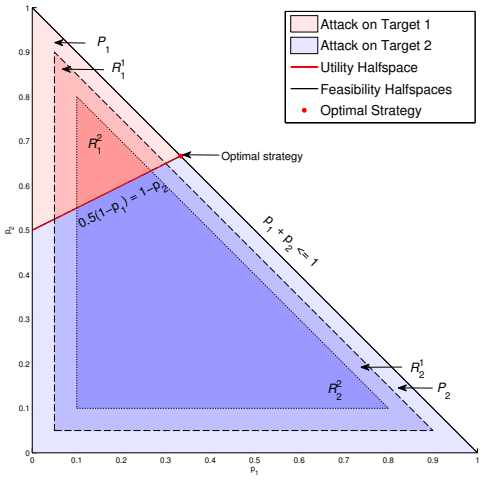
An important requirement for many optimization algorithms, including the one developed by [170], is having a "well-centered" initial feasible point in the region of optimization. There are two challenges involved in discovering an initial feasible point in the interior of every region. First, establishing that a region is non-empty, possibly by finding a boundary point. Second, obtaining a point that has a significant margin from the boundary. We carry out these tasks by executing the optimization in a hierarchy of sets where at each level the optimization task only considers a subset of the targets and the feasibility space. We then show that optimization in one level of this hierarchy helps us find initial points in new regions that are well-centered in higher levels of the hierarchy.

To this end, let us define *restricted regions*. These regions are obtained by first perturbing the defining half-spaces of $\mathcal{P}$ so that they conform to a given representation length, and then trimming the boundaries by a given width (See Figure 2.1).

In the remainder of this chapter, we use $\gamma = \frac{1}{(n+1)2^{L+1}}$ to denote the accuracy of the representation and the width of the trimming procedure for obtaining restricted regions. More precisely:

Definition 2.4.5 (restricted regions)**.** *The set $\mathcal{R}^k \in \mathbb{R}^n$ is defined by the intersection of the following half-spaces: For all i, $(\mathbf{e}_i, k\gamma)$. For all $(\mathbf{h}^*, b^*) \in H_{\mathcal{P}}$, a half-space $(\mathbf{h}, b + k\gamma)$, such that $\mathbf{h} = \gamma \lfloor \frac{1}{\gamma} \mathbf{h}^* \rfloor$ and $b = \gamma \lceil \frac{1}{\gamma} b^* \rceil$. Furthermore, for every $i \in \mathcal{N}$, define $\mathcal{R}^k_i = \mathcal{R}^k \cap \mathcal{P}_i$.*

The next Lemma shows that the restricted regions are subsets of the feasibility space, so, we can make best-response queries within them.



Target	Attacker	Defender
1	$0.5(1-p_1)$	$-0.5(1-p_1)$
2	$(1-p_2)$	$-(1-p_2)$

Figure 2.1: The game payoff table on the right and optimization regions on the left. A security game with one resource that can cover one of two targets. The attacker receives utility 0.5 from attacking target 1 and utility 1 from attacking target 2, when they are not defended; he receives 0 utility from attacking a target that is being defended. The defender's utility is the zero-sum complement.

Lemma 2.4.6. *For any $k \geq 0$, $\mathcal{R}^k \subseteq \mathcal{P}$.*

Proof. Let $\mathbf{p} \notin \mathcal{P}$, by Lemma 2.4.4 one of the following cases holds: (1) There exists i, such that $\mathbf{e}_i \cdot \mathbf{p} < 0$. In this case, $\mathbf{e}_i \cdot \mathbf{p} \leq k\gamma$. So, $\mathbf{p} \notin R^k$. (2) There exists $(\mathbf{h}^*, b^*) \in H_{\mathcal{P}}$ such that $\mathbf{h}^* \cdot \mathbf{p} < b^*$. Let $(\mathbf{h}, b)$ be the corresponding half-space of $(\mathbf{h}^*, b^*)$ in $\mathcal{R}^k$. Then,

$$\mathbf{h} \cdot \mathbf{p} = \mathbf{h}^* \cdot \mathbf{p} + (\mathbf{h} - \mathbf{h}^*) \cdot \mathbf{p} < b^* + (\mathbf{h} - \mathbf{h}^*) \cdot \mathbf{p} < b$$

where the last transition is by the fact that $\mathbf{h} \preceq \mathbf{h}^*$, $b > b^*$, and $\mathbf{p} \succeq \mathbf{0}$. □

The next two lemmas show that in $\mathcal{R}^k$ one can reduce each coverage probability individually down to $k\gamma$, and the optimal conservative strategy in $\mathcal{R}^k$ indeed reduces the coverage probabilities of all targets outside the best-response set to $k\gamma$.

Lemma 2.4.7. *Let $\mathbf{p} \in \mathcal{R}^k$, and let $\mathbf{q}$ such that $k\boldsymbol{\gamma} \preceq \mathbf{q} \preceq \mathbf{p}$. Then $\mathbf{q} \in \mathcal{R}^k$.*

Proof. If $\mathcal{R}^k = \emptyset$ is empty then the result holds trivially. If $\mathcal{R}^k \neq \emptyset$, then there exists $\mathbf{p} \succeq k\boldsymbol{\gamma}$ such that $\mathbf{p} \in \mathcal{R}^k$. By Lemmas 2.4.3 and 2.4.4, $\mathcal{P}$ has half-spaces of the form $(\mathbf{e}_i, 0)$ and $(\mathbf{h}^*, b^*)$, such that $\mathbf{h}^* \preceq \mathbf{0}$ and $b^* \leq 0$. By construction of $\mathcal{R}^k$, we have half-spaces of the form $(\mathbf{e}_i, k\gamma)$ for all i, and half-spaces $(\mathbf{h}, b + k\gamma)$ such that $\mathbf{h} = \lfloor \frac{1}{\gamma}\mathbf{h}^* \rfloor \preceq \mathbf{h}^*$. Since, $k\boldsymbol{\gamma} \preceq \mathbf{p} \in \mathcal{R}^k$, $\mathbf{h} \cdot k\boldsymbol{\gamma} = \|h\|_1 k\gamma \geq b$. So, $b \leq -k\gamma$. The proof is completed by the fact that the conditions of Lemma 2.4.4 hold. □

Lemma 2.4.8. *Let $\mathbf{s}$ and its corresponding coverage probability $\mathbf{p}$ be a conservative optimal strategy in $\mathcal{R}^k$. Let $i^* = b(\mathbf{s})$ and $B = \{i : U_a(i, p_i) = U_a(i^*, p_{i^*})\}$. Then for any $i \notin B$, $p_i = k\gamma$.*

Proof. Note that there is a positive gap between the attacker's payoff from attacking a best-response target versus another target, i.e. $\min_{i \notin B} U_a(i^*, p_{i^*}) - U_a(i, p_i) > 0$. Since U_a is continuous and decreasing in the coverage probability, for any $i \notin B$, if $p_i > k\gamma$ there exists $0 < \delta \leq p_i - k\gamma$ such that $U_a(i^*, p_{i^*}) > U_a(i, p_i - \delta)$. Let $\mathbf{q}$ be defined such that for all $i \notin B$, $q_i = p_i - \delta \geq k\gamma$, and for all $i \in B$, $q_i = p_i$. Then by Lemma 2.4.7, $\mathbf{q}$ is implementable by some strategy $\mathbf{s}_q$ in $\mathcal{R}^k$. Furthermore, $b(\mathbf{q}) = i^*$, so, $\mathbf{s}_q$ is also an optimal strategy. This contradicts the fact that $\mathbf{s}$ is a conservative optimal strategy. □

The following Lemma shows that if every non-empty $\mathcal{P}_i$ contains a large enough ball, then $\mathcal{R}_i^n \neq \emptyset$.

Lemma 2.4.9. *For any $i \in \mathcal{N}$ and $k \leq n$ such that $\mathcal{P}_i$ contains a ball of radius $r > \frac{1}{2^L}$, $\mathcal{R}_i^k \neq \emptyset$.*

Proof. Let $\mathbf{p}$ be the center of the ball. Then for any j, $p_j \geq r \geq k\gamma$, so $\mathbf{e}_j \cdot \mathbf{p} \geq k\gamma$. For any $(\mathbf{h}, b)$ defining half-space of $\mathcal{R}^k$ and its corresponding half-space $(\mathbf{h}^*, b^*)$ of $\mathcal{P}$ we have:

$$\mathbf{h} \cdot \mathbf{p} - b = \mathbf{h}^* \mathbf{p} - b^* + (\mathbf{h} - \mathbf{h}^*) \cdot \mathbf{p} + (b^* - b) \geq r - \gamma n - (\gamma + k\gamma) \geq 0,$$

where the second transition is by the fact that $\mathbf{h}^* \mathbf{p} = r$ is the margin of $\mathbf{p}$ from a normalized half-space $\mathbf{h}^*$, and that for any value x and $1/\gamma \in \mathbb{Z}$, $\gamma \lfloor \frac{1}{\gamma} x \rfloor$ and $\gamma \lceil \frac{1}{\gamma} x \rceil$ are within γ from x. Hence, $\mathbf{p} \in \mathcal{R}_i^k$. □

The next lemma provides the main insight behind our search for the region with the highest-paying optimal strategy. It implies that we can restrict our search to strategies that are optimal for a subset of targets in $\mathcal{R}^k$, if the attacker also agrees to play within that subset of targets. At any point, if the attacker chooses a target outside the known regions, he is providing us with a point in a new region. Crucially, Lemma 2.4.10 requires that we optimize exactly inside each restricted region, and we show below (Algorithm 2.1 and Lemma 2.4.13) that this is indeed possible.

Lemma 2.4.10. *Assume that for every i, if $\mathcal{P}_i$ is non-empty, then it contains a ball of radius $\frac{1}{2^L}$. Given $K \subseteq \mathcal{N}$ and $k \leq n$, let $\mathbf{p} \in \mathcal{R}^k$ be the coverage probability of the strategy that has $k\gamma$ probability mass on targets in $\mathcal{N} \setminus K$ and is optimal if the attacker were to be restricted to attacking targets in K. Let $\mathbf{p}^*$ be the optimal strategy in $\mathcal{P}$. If $b(\mathbf{p}) \in K$ then $b(\mathbf{p}^*) \in K$.*

Proof. Assume on the contrary that $b(\mathbf{p}^*) = i^* \notin K$. Since $\mathcal{P}_{i^*} \neq \emptyset$, by Lemma 2.4.9, there exists $\mathbf{p}' \in \mathcal{R}_{i^*}^k$.

For ease of exposition, replace $\mathbf{p}$ with its corresponding conservative strategy in $\mathcal{R}^k$. Let B be the set of targets that are tied for the attacker's best-response in $\mathbf{p}$, i.e. $B = \arg\max_{i \in \mathcal{N}} U_a(i, p_i)$. Since $b(\mathbf{p}) \in K$ and ties are broken in favor of the "best" target, i.e. i^*, it must be that $i^* \notin B$. Then, for any $i \in B$, $U_a(i, p_i) > U_a(i^*, k\gamma) \geq U_a(i^*, p'_{i^*}) \geq U_a(i, p'_i)$. Since U_a is decreasing in the coverage probability, for all $i \in B$, $p'_i > p_i$. Note that there is a positive gap between the attacker's payoff for attacking a best-response target versus another target, i.e. $\Delta = \min_{i' \in K \setminus B, i \in B} U_a(i, p_i) - U_a(i', p_{i'}) > 0$, so it is possible to increase p_i by a small amount without changing the best response. More precisely, since U_a is continuous and decreasing in the coverage probability, for every $i \in B$, there exists $\delta < p'_i - p_i$ such that for all $i' \in K \setminus B$, $U_a(i', p_{i'}) < U_a(i, p'_i - \delta) < U_a(i, p_i)$.

Let $\mathbf{q}$ be such that for $i \in B$, $q_i = p'_i - \delta$ and for $i \notin B$, $q_i = p_i = k\gamma$ (by Lemma 2.4.8 and the fact that $\mathbf{p}$ was replaced by its conservative equivalent). By Lemma 2.4.7, $\mathbf{q} \in \mathcal{R}^k$. Since for all $i \in B$ and $i' \in K \setminus B$, $U_a(i, q_i) > U_a(i', q_{i'})$, $b(\mathbf{q}) \in B$. Moreover, because U_d is increasing in the coverage probability for all $i \in B$, $U_d(i, q_i) > U_d(i, p_i)$. So, $\mathbf{q}$ has higher payoff for the defender when the attacker is restricted to attacking K. This contradicts the optimality of $\mathbf{p}$ in $\mathcal{R}^k$. Therefore, $b(\mathbf{p}^*) \in K$. □

If the attacker attacks a target i outside the set of targets K whose regions we have already discovered, we can use the new feasible point in $\mathcal{R}^k_i$ to obtain a well-centered point in $\mathcal{R}^{k-1}_i$, as the next lemma formally states.

Lemma 2.4.11. *For any k and i, let $\mathbf{p}$ be any strategy in $\mathcal{R}^k_i$. Define $\mathbf{q}$ such that $q_i = p_i - \frac{\gamma}{2}$ and for all $j \neq i$, $q_j = p_j + \frac{\gamma}{4\sqrt{n}}$. Then, $\mathbf{q} \in \mathcal{R}^{k-1}_i$ and $\mathbf{q}$ has distance $\frac{\gamma}{2n}$ from the boundaries of $\mathcal{R}^{k-1}_i$.*

Proof. The boundaries of $\mathcal{R}^k$ are defined by $(\mathbf{e}_i, k\gamma)$ for all $i \in \mathcal{N}$ and a half-space $(\mathbf{h}, b + k\gamma)$ for every half-space $(\mathbf{h}^*, b^*) \in H_{\mathcal{P}}$ such that $\mathbf{h} = \gamma\lfloor\frac{1}{\gamma}\mathbf{h}^*\rfloor$ and $b = \gamma\lceil\frac{1}{\gamma}b^*\rceil$. In addition $\mathcal{R}^k_i$ is the intersection of $\mathcal{R}^k$ with half-spaces $U_i(i, p_i) \geq U(i', p_{i'})$ for all $i' \neq i$. Let $\mathrm{dist}(\cdot)$ denote the signed distance of a point from a half-space. For every $j \neq i$,

$$\begin{aligned}\mathrm{dist}(\mathbf{q}, (\mathbf{e}_j, (k-1)\gamma)) &= \frac{\mathbf{e}_j \cdot \mathbf{q} - (k-1)\gamma}{\|\mathbf{e}_j\|_2} = q_j - (k-1)\gamma \geq p_j + \frac{\gamma}{4\sqrt{n}} - (k-1)\gamma \\ &\geq k\gamma + \frac{\gamma}{4\sqrt{n}} - (k-1)\gamma > \frac{\gamma}{2n}.\end{aligned}$$

Moreover,

$$\begin{aligned}\mathrm{dist}(\mathbf{q}, (\mathbf{e}_i, (k-1)\gamma)) &= \frac{\mathbf{e}_i \cdot \mathbf{q} - (k-1)\gamma}{\|\mathbf{e}_i\|_2} = q_i - (k-1)\gamma = p_i - \frac{\gamma}{2} - (k-1)\gamma \\ &\geq k\gamma - \frac{\gamma}{2} - (k-1)\gamma \geq \frac{\gamma}{2n}.\end{aligned}$$

Finally, for every $(\mathbf{h}, b + (k-1)\gamma)$,

$$\begin{aligned}\mathrm{dist}(\mathbf{q}, (\mathbf{h}, b + (k-1)\gamma)) &= \frac{\mathbf{h} \cdot \mathbf{q} - (b + (k-1)\gamma)}{\|\mathbf{h}\|_2} \geq \frac{\mathbf{h} \cdot \mathbf{p} + h \cdot (\mathbf{q} - \mathbf{p}) - (b + (k-1)\gamma)}{2} \\ &\geq \frac{h \cdot (\mathbf{q} - \mathbf{p}) + \gamma}{2} \geq -\frac{\gamma}{8\sqrt{n}}\|\mathbf{h}\|_1 - h_i\left(\frac{\gamma}{4} + \frac{\gamma}{8\sqrt{n}}\right) + \frac{\gamma}{2} \\ &\geq -\frac{\gamma}{8\sqrt{n}}\|\mathbf{h}\|_1 + \frac{\gamma}{2} \geq -\frac{\gamma}{8\sqrt{n}}(\sqrt{n} + n\gamma) + \frac{\gamma}{2} \\ &\geq \frac{3\gamma}{8} - \frac{\sqrt{n}\gamma^2}{8} \geq \frac{\gamma}{2n},\end{aligned}$$

where the first inequality is by the fact that $\|\mathbf{h}\|_2 \leq \|\mathbf{h}^*\|_2 + \gamma\sqrt{n} \leq 1 + \gamma\sqrt{n} < 2$ (by the triangle inequality), the penultimate inequality is by the fact that $\|\mathbf{h}\|_1 \leq \|\mathbf{h}^*\|_1 + n\gamma \leq \sqrt{n}\|\mathbf{h}^*\|_2 + n\gamma$, and the last inequality follows from $\gamma = \frac{1}{(n+1)2^{L+1}} < \frac{1}{n\sqrt{n}}$.

As for the utility half-spaces of the form $U_a(i, q_i) - U_a(i', q_{i'}) \geq 0$, for every i', the probability $\mathbf{q}$ has moved away by at least $\min(\frac{\gamma}{2}, \frac{\gamma}{4\sqrt{n}}) \geq \frac{\gamma}{2n}$ from every half-space. Moreover, by reducing the coverage probability on the attacked target and increasing it on other targets, the attacker receives even larger payoff from attacking i, so $\mathbf{q}$ still induces an attack on i, i.e. $\mathbf{q} \in \mathcal{P}_i$. Finally, the signed distance of $\mathbf{q}$ from every half-space is greater than $\frac{\gamma}{2n}$, therefore $\mathbf{q} \in \mathcal{R}^{k-1} \cap \mathcal{P}_i = \mathcal{R}_i^{k-1}$, and it has distance $\frac{\gamma}{2n}$ from the boundaries of $\mathcal{R}_i^{k-1}$.

□

2.4.3 An Oracle for the Convex Region

We use a three-step procedure for defining a membership oracle for $\mathcal{P}$ or $\mathcal{R}_i^k$. Given a vector $\mathbf{p}$, we first use the half-space representation of $\mathcal{P}$ (or $\mathcal{R}^k$) described in Section 2.4.1 to determine whether $\mathbf{p} \in \mathcal{P}$ (or $\mathbf{p} \in \mathcal{R}^k$). We then find a strategy $\mathbf{s}$ that implements $\mathbf{p}$ by solving a linear system with constraints $M\mathbf{s}^\top = \mathbf{p}^\top$, $\mathbf{0} \preceq \mathbf{s}$, and $\|\mathbf{s}\|_1 = 1$. Lastly, we make a best-response query to the attacker for strategy $\mathbf{s}$. If the attacker responds by attacking i, then $\mathbf{p} \in \mathcal{P}_i$ (or $\mathbf{p} \in \mathcal{R}_i^k$), else $\mathbf{p} \notin \mathcal{P}_i$ (or $\mathbf{p} \notin \mathcal{R}_i^k$).

2.4.4 The Algorithms

In this section, we define algorithms that use the results from previous sections to prove Theorem 2.4.1. First, we define Algorithm 2.1, which receives an approximately optimal strategy in $\mathcal{R}_i^k$ as input, and finds the optimal strategy in $\mathcal{R}_i^k$. As noted above, obtaining exact optimal solutions in $\mathcal{R}_i^k$ is required in order to apply Lemma 2.4.10, thereby ensuring that we discover new regions when lucrative undiscovered regions still exist.

Algorithm 2.1: LATTICE-ROUNDING

1: Input: $\mathbf{p}$, an $\frac{1}{2^{6n(L+1)}}$-approximate optimal strategy in $\mathcal{R}_i^k$. A best-response oracle for $\mathcal{R}_i^k$.
2: **for** $j \neq i$ **do**
3: Make best-response queries to binary search for the smallest $p'_j \in [k\gamma, p_j]$ to accuracy $\frac{1}{2^{5n(L+1)}}$, such that $i = b(\mathbf{p}')$, where for all $j' \neq j$, $p'_{j'} \leftarrow p_{j'}$.
4: **end for**
5: **for** $j \neq i$ **do**
6: Set r_j and q_j respectively to the smallest and second smallest rational numbers with denominator at most $2^{2n(L+1)}$, that are larger than $p'_j - \frac{1}{2^{5n(L+1)}}$.
7: **end for**
8: Let $\mathbf{p}^*$ be such that p_i^* is the unique rational number with denominator at most $2^{2n(L+1)}$ in $[p_i, p_i + \frac{1}{2^{4n(L+1)}})$ (Refer to the proof for uniqueness), and for all $j \neq i$, $p_j^* \leftarrow r_j$.
9: Query $x \leftarrow b(\mathbf{p}^*)$.
10: **if** $x \neq i$, **then** $p_x^* \leftarrow q_x$ and go to step 9.
11: **return** $\mathbf{p}^*$.

The next two Lemmas establish the guarantees of Algorithm 2.1. The first is a variation of a well-known result in linear programming [127] that is adapted specifically for our problem

setting.

Lemma 2.4.12. *Let $\mathbf{p}^*$ be a basic optimal strategy in $\mathcal{R}_i^k$, then for all i, p_i^* is a rational number with denominator at most $2^{2n(L+1)}$.*

Proof. Let $\mathcal{R}_i^k$ be represented as a system $\{\mathbf{p} : A\mathbf{p}^\top \succeq \mathbf{b}\}$ where there is a row (constraint) for each half-space that defines $\mathcal{R}^k$, and a row for each $i' \neq i$ of the form $U_a(i, p_i) - U_a(i', p_{i'}) \geq 0$. Furthermore, assume that A is normalized so that every row has integral coefficients. Note that by the definition of the game representation length, each coefficient of the utility rows is at most 2^L. Moreover, by the definition of the defining half-spaces of $\mathcal{R}^k$, each coefficients of the feasibility constraints are at most $(n+1)2^{L+1}$.

We know that each basic solution to the above LP is at the intersection of n independent constraints of A. Let $\mathbf{p}^*$ be such a solution. Let D represent those n hyper-planes. Using Cramer's rule, for all i, $p_i^* = \frac{\det(D_i)}{\det(D)}$, where the D_i is D with its i^{th} column replaced by b. Using Hadamard's inequality,

$$\det(D) \leq \prod_{i=1}^{n} \sqrt{\sum_{j=1}^{n} d_{ij}^2} \leq \prod_{i=1}^{n} (n+1)2^{L+1}\sqrt{n} \leq n^{2n}2^{n(L+1)} \leq 2^{2n(L+1)},$$

Where the last inequality is by the fact that $L > n \log n$. □

Lemma 2.4.13. *For any k and i, let $\mathbf{p}$ be a $\frac{1}{2^{6n(L+1)}}$-approximate optimal strategy in $\mathcal{R}_i^k$. Algorithm 2.1 finds the optimal strategy in $\mathcal{R}_i^k$ in $O(nL)$ best-response queries.*

Proof. Let $\mathbf{p}^*$ be the optimal strategy in $\mathcal{R}_i^k$. By Lemma 2.4.12, for all j, p_j^* has a denominator of at most $2^{2n(L+1)}$. Note that the difference between two distinct rational numbers with denominators at most $2^{2n(L+1)}$ is at least $\frac{1}{2^{4n(L+1)}}$.

Strategy $\mathbf{p}$ is a $\frac{1}{2^{6n(L+1)}}$-approximate optimal strategy and the utilities have representation length of at most L, so

$$p_i^* - p_i \leq 2^L \cdot \frac{1}{2^{6n(L+1)}} < \frac{1}{2^{4n(L+1)}}.$$

Therefore, $p_i^* \in [p_i, p_i + \frac{1}{2^{4n(L+1)}})$. Since this range is smaller than the difference between two rational numbers with denominator at most $2^{2n(L+1)}$, there is at most one such rational number in this range, to which our algorithms sets p_i^*. Note that the absence of such a rational number is contradictory to Lemma 2.4.12 or the fact that $\mathbf{p}$ was a $\frac{1}{2^{6n(L+1)}}$-approximate optimal strategy.

For all j, let $p'_j \geq k\gamma$ be the smallest coverage probability, with accuracy $\frac{1}{2^{5n(L+1)}}$, such that $U_a(i, p_i) \geq U_a(j, p'_j)$. Then, $p_j^* \geq p'_j - \frac{1}{2^{5n(L+1)}}$. Let r_j and q_j, respectively, be the smallest and second smallest rational numbers with denominator at most $2^{2n(L+1)}$ in the range $[p'_j - \frac{1}{2^{5n(L+1)}}, 1)$. We claim that $p_j^* = r_j$ or q_j. To prove this claim, it is sufficient to show that for all j, $U_a(j, q_j) \leq U_a(i, p_i^*)$. Since q_i is the second smallest rational number with denominator at most $2^{2n(L+1)}$ in the given range, then $q_j \geq p'_j - \frac{1}{2^{5n(L+1)}} + \frac{1}{2^{4n(L+1)}} > \frac{1}{2^{4n(L+1)+1}}$. Then,

$$U_a(j, p'_j) - U_a(j, q_j) \geq \frac{1}{2^L}(q_j - p'_j) \geq \frac{1}{2^{4n(L+1)+L+1}}$$

Algorithm 2.2: Optimize Defender Strategy

1: Input: Accuracy level ϵ, confidence δ, and best-response oracle $b(\cdot)$.
2: $\gamma \leftarrow \frac{1}{(n+1)2^{L+1}}$, $\delta' \leftarrow \frac{\delta}{n^2}$, and $k \leftarrow n$.
3: Query $i \leftarrow b(k\boldsymbol{\gamma})$.
4: Let $K \leftarrow \{i\}$, $X \leftarrow \{\mathbf{x}^i\}$, where $x^i_i = k\gamma - \gamma/2$ and for $j \neq i$, $x^i_j = k\gamma + \frac{\gamma}{4\sqrt{n}}$.
5: **for** $i \in K$ **do**
6: **if** between line 11 to 13 $i' \notin K$ is attacked as a response to some strategy $\mathbf{p}$ **then**
7: Let $x^{i'}_{i'} \leftarrow p_{i'} - \gamma/2$ and for $j \neq i'$, $x^{i'}_j \leftarrow p_j + \frac{\gamma}{4\sqrt{n}}$.
8: $X \leftarrow X \cup \{\mathbf{x}^{i'}\}$, $K \leftarrow K \cup \{i'\}$, and $k \leftarrow k - 1$.
9: Restart the loop at step 5.
10: **end if**
11: Use Theorem 2.3.1 with set of targets K. With probability $1 - \delta'$ find a $\mathbf{q}^i$ that is a $\frac{1}{2^{6n(L+1)}}$-approximate optimal strategy restricted to set K.
12: Use Algorithm 2.1 on $\mathbf{q}^i$ restricted to set of targets $j \in K$ to find the optimal strategy $\mathbf{q}^{*i}$ in $\mathcal{R}^k_i$. For all $j \notin K$, let $q^{*i}_j \leftarrow k\gamma$.
13: Query $b(\mathbf{q}^{*i})$.
14: **end for**
15: **for** all $i \in K$ **do**
16: Using Theorem 2.3.1 find $\mathbf{p}^{*i}$ that is an ϵ-approximate strategy in $\mathcal{P}_i$
17: **end for**
18: **return** $\mathbf{p}^{*i}$ that has the highest payoff to the defender.

$$
\begin{aligned}
&> \frac{2^L}{2^{6n(L+1)}} \geq 2^L \left(U_d(i, p^*_i) - U_d(i, p_i)\right) \\
&> U_a(i, p_i) - U_a(i, p^*_i).
\end{aligned}
$$

Since, $U_a(i, p_i) \geq U_a(j, p'_j)$, the above inequality implies that $U_a(j, q_j) \leq U_a(i, p^*_i)$. So, for each j, it is sufficient to query the attacker to see whether $U_a(j, r_j) \leq U_a(i, p^*_i)$ if so then $p^*_j = r_j$, else $p^*_j = q_j$.

For each j, this algorithm makes $O(\log 2^{5(L+1)}) = O(L)$ queries to find p'_j with accuracy $\frac{1}{2^{5n(L+1)}}$. Values of p^*_i, r_j and q_j are computed without any best-response queries. Because $p^*_j = r_j$ or q_j, step 9, is repeated at most n times, so there are n additional queries. In conclusion, our algorithm makes $O(nL + n) = O(nL)$ many queries in total. □

At last, we are ready to prove our main result, which provides guarantees for Algorithm 2.2, given below.

Theorem 2.4.1 (restated). *Consider a security game with n targets and representation length L, such that for every target, the set of implementable coverage probability vectors that induce an attack on that target, if non-empty, contains a ball of radius $1/2^L$. For any $\epsilon, \delta > 0$, with probability $1 - \delta$, Algorithm 2.2 finds a defender strategy that is optimal up to an additive term of ϵ, using $O(n^{6.5}(\log \frac{n}{\epsilon\delta} + L))$ best-response queries to the attacker.*

Proof Sketch. For each $K \subseteq \mathcal{N}$ and k, the loop at step 5 of Algorithm 2.2 finds the optimal strategy if the attacker was restricted to attacking targets of K in $\mathcal{R}^k$.

Every time the "if" clause at step 6 is satisfied, the algorithm expands the set K by a target i' and adds $\mathbf{x}^{i'}$ to the set of initial points X, which is an interior point of $\mathcal{R}_{i'}^{k-1}$ (by Lemma 2.4.11). Then the algorithm restarts the loop at step 5. Therefore every time the loop at step 5 is started, X is a set of initial points in K that have margin $\frac{\gamma}{2n}$ in $\mathcal{R}^k$. This loop is restarted at most $n-1$ times.

We reach step 15 only when the best-response to the optimal strategy that only considers targets of K is in K. By Lemma 2.4.10, the optimal strategy is in $\mathcal{P}_i$ for some $i \in K$. By applying Theorem 2.3.1 to K, with an oracle for $\mathcal{P}$ using the initial set of point X which has $\gamma/2n$ margin in $\mathcal{R}^0$, we can find the ϵ-optimal strategy with probability $1-\delta'$. There are at most n^2 applications of Theorem 2.3.1 and each succeeds with probability $1-\delta'$, so our overall procedure succeeds with probability $1-n^2\delta' \geq 1-\delta$.

Regarding the number of queries, every time the loop at step 5 is restarted $|K|$ increases by 1. So, this loop is restarted at most $n-1$ times. In a successful run of the loop for set K, the loop makes $|K|$ calls to the algorithm of Theorem 2.3.1 to find a $\frac{1}{2^{6n(L+1)}}$-approximate optimal solution. In each call, X has initial points with margin $\frac{\gamma}{2n}$, and furthermore, the total feasibility space is bounded by a sphere of radius $\sqrt{n}$ (because of probability vectors), so each call makes $O(n^{4.5}(\log \frac{n}{\delta} + L))$ queries. The last call looks for an ϵ-approximate solution, and will take another $O(n^{4.5}(\log \frac{n}{\epsilon\delta} + L))$ queries. In addition, our the algorithm makes n^2 calls to Algorithm 2.1 for a total of $O(n^3 L)$ queries. In conclusion, our procedure makes a total of $O(n^{6.5}(\log \frac{n}{\epsilon\delta} + L)) = \text{poly}(n, L, \log \frac{1}{\epsilon\delta})$ queries. □

2.5 Discussion

Our main result focuses on the query complexity of our problem. We believe that, indeed, best response queries are our most scarce resource, and it is therefore encouraging that an (almost) optimal strategy can be learned with a polynomial number of queries.

It is worth noting, though, that some steps in our algorithm are computationally inefficient. Specifically, our membership oracle needs to determine whether a given coverage probability vector is implementable. We also need to explicitly compute the feasibility half-spaces that define $\mathcal{P}$. Informally speaking, (worst-case) computational inefficiency is inevitable, because computing an optimal strategy to commit to is computationally hard even in simple security games [183].

Nevertheless, deployed security games algorithms build on integer programming techniques to achieve satisfactory runtime performance in practice [256]. These algorithms often focus on efficiently solving the following sub-problem in Stackelberg security games: Given a coverage probability vector $\mathbf{p}$, find a succinct representation of the mixed strategy $\mathbf{s}$ that implements it. This is exactly the step we use to determine whether and how a coverage probability vector is implementable. Therefore, an interesting open question is whether we can learn a highly accurate defender strategy if we have access to such an algorithm that efficiently maps the coverage probability space to the mixed strategy space. That is, is there a way around explicitly computing feasibility half-spaces that define $\mathcal{P}$ in our algorithm? Solving this problem would allow us to

use deployed tools from the security game domain and obtain truly practical learning algorithms for dealing with payoff uncertainty in security games.

Chapter 3

Learning about a Boundedly Rational Attacker in Stackelberg Games

3.1 Introduction

In this chapter, we consider Stackelberg security games in settings where the attacker may not be fully rational. In particular, we consider the recent application of Stackelberg security games to protecting wildlife, fisheries and forests. This *green security game* (GSG) research [116, 213] differs from earlier work in SSGs applied to counter-terrorism [223] in two ways: first, poacher behavior has been observed to deviate from fully strategic behavior, perhaps due to poacher's lack of sophisticated surveillance methods; second, there is significant historical data available in GSGs (e.g., wildlife crime, arrests of poachers) from repeated defender-attacker interactions.

Given the availability of the data in this domain, machine learning has begun to play a critical role in improving defender resource allocation in GSGs, taking the place of domain experts as the leading method for estimating attacker utilities and preferences. Indeed, inspired by GSGs, researchers have focused on learning attacker bounded rationality models and designing optimal defender strategies against such attackers [249, 271]. While existing results are encouraging, as we explain in detail below, a shortcoming of the state-of-the-art approach is that its effectiveness implicitly relies on the properties of the underlying distribution from which data is obtained—if near-optimal strategies have not already been played in the past, a near-optimal strategy cannot be learned.

3.1.1 Our Results

We take the next step in developing the theory and practice of learning attacker behavior. Our first contribution is a theoretical analysis of the learnability of (a generalization of) the most well-studied bounded rationality attacker behavior model in SSGs: Subjective Utility Quantal Response (SUQR) [214], which is a parametric specification of the attacker's behavior. We find, perhaps surprisingly, that if the data contains a polynomial number of attacker responses to each of only *three* defender strategies that are sufficiently different from each other (precise statement is given in Theorem 3.3.1), then we can learn the model parameters with high accuracy.

Qualitatively, this means that we can expect to learn an excellent strategy from real-world historical data, as typically each defender strategy is played repeatedly over a period of time. It also means that, even if we collect additional data in the future by playing whatever strategy seems good based on existing data, the new data will quickly lead to an *optimal* strategy, under the very mild assumption that the new strategies that are played are somewhat different from the previous ones.

Building on our analysis of the generalized SUQR model, as part of our second contribution, we analyze the learnability of the more general class of attacker behavior models specified by (non-parametric) Lipschitz functions. This is an extremely expressive class, and therefore, naturally, learning an appropriate model requires more data. Even for this class we can learn the attacker response function with high accuracy with a polynomial number of defender strategies—but we make more stringent assumptions regarding how these strategies are selected. Our analysis works by approximating Lipschitz functions using polynomial functions.

3.1.2 Related Work

Our work is most closely related to the recent paper of Sinha et al. [249]. They learn to predict attacker responses in the PAC model of learning. Crucially, following the PAC model, the dataset is assumed to be constructed by drawing defender strategies (and attacker responses) from a fixed but unknown distribution—and the accuracy of the outcome of the learning process is then measured *with respect to that same distribution*. In particular, if the training data is concentrated in suboptimal regions of the defender strategy space, the PAC model approach allows accurate prediction of attacker responses in those regions, but may not help pinpoint a globally optimal strategy (as discussed at length by Sinha et al. [249]). In contrast, our approach leads to *uniformly* accurate prediction, in the sense that we can accurately predict the attacker responses to *any* defender strategy, even if we only observe suboptimal strategies. As we show, this provides sufficient information to identify a near-optimal strategy.

Another related line of work, as presented in Chapter 2 and related works [37, 56, 190], explores an adaptive learning approach, where an optimal strategy is learned by adaptively playing defender strategies and observing the attacker responses. That approach cannot make use of historical data. Moreover these works may play severely suboptimal strategies in order to most efficiently pinpoint the optimal strategy. In contrast, in the adaptive interpretation of the work in this chapter, we can learn an optimal strategy by playing *any* three (sufficiently different) strategies, including ones that seem optimal based on existing evidence. Interestingly, the reason why we are able to do so much better is that we assume a bounded rational attacker that responds probabilistically to defender strategies, while the foregoing papers assume a perfectly rational attacker. At first sight, it may appear that learning in the bounded rationality model may be harder since the behavior of the attacker is random. But by repeatedly playing the same strategy, we gain information about the attacker's utility for all targets, whereas in the perfectly rational case, to gain information about new targets, the learning algorithm has to discover the "best response regions" of those targets.

3.2 Preliminaries

We consider a variant of the Stackelberg security game model in which the attacker does not act rationally. We consider a set $\mathcal{N} = \{1, \ldots, n\}$ of n *targets*. We directly work in the *coverage probability space*, where the set of defender mixed strategies is the set of vectors $\mathcal{P} \subseteq [0,1]^n$ and for each $\mathbf{p} \in \mathcal{P}$, p_i represents the probability with which the defender protects target i in mixed strategy $\mathbf{p}$. As discussed in earlier chapters, this set is determined by defender's *security resources* and the subsets of targets that each resource can defend simultaneously. A *pure deployment* of the defender is then an assignment of resources to targets, and a *mixed deployment* of the defender is a distribution over pure deployments. In this context, the coverage probability vector induced by a mixed deployment is defined as the probability with which each target is defended under this mixed deployment. As we shall see, the behavior and utility of the attacker only depend on the defender's choice of coverage probabilities, and therefore we choose to represent the defender's action space by the set of coverage probabilities $\mathcal{P}$.

As before, we use $u_d(i, p_i)$ and $u_a(i, p_i)$ to denote defender and attacker utilities when the attacker attacks target i under mixed strategy $\mathbf{p}$. To simplify these notations, in this chapter, we use $v_i(p_i)$ and $u_i(p_i)$ to denote the defender and attacker utilities, respectively. Previous work on learning in security games has mainly focused on utilities that are linear functions of the coverage probability (like Chapter 2), or linear functions with the additional constraint that $u_i(p_i) = wp_i + c_i$ in the generalized SUQR model of Sinha et al. [249]. Our main result pertains to (unrestricted) linear utility functions, but later we also deal with higher degree polynomials.

Upon observing the defender's strategy $\mathbf{p}$, the attacker computes the utility on each target i, $u_i(p_i)$, and based on these utilities *responds* to the defender's strategy. In this chapter, we consider a non-adaptive attacker who attacks target i with probability

$$D^{\mathbf{p}}(i) = \frac{e^{u_i(p_i)}}{\sum_{j \in \mathcal{N}} e^{u_j(p_j)}}. \tag{3.1}$$

This model corresponds to the *Luce* model from quantal choice theory [198, 206], and is a special case of the *logit quantal response* model. Together with our choice of utility functions, our model is a generalization of bounded rationality models considered in previous work, such as SUQR [214] and generalized SUQR [249].

Suppose the same mixed strategy $\mathbf{p}$ is played for multiple time steps. We denote the empirical distribution of attacks on target i under $\mathbf{p}$ by $\hat{D}^{\mathbf{p}}(\cdot)$. Furthermore, we assume that for the strategies considered in our work, and for all i, $D^{\mathbf{p}}(i) \geq \rho$ for some $\rho = 1/\text{poly}(n)$. This assumption is required to estimate the value of $D^{\mathbf{p}}(i)$ with polynomially many samples.

Our goal is to learn the utility functions, $u_i(\cdot)$ for all $i \in \mathcal{N}$, by observing attacker's responses to a choice of coverage probability vectors $\mathbf{p} \in \mathcal{P}$. This allows us to find an approximately optimal defender strategy—the strategy that leads to the best defender utility. We say that $\hat{u}_i : [0,1] \to \mathbb{R}$ *uniformly approximates* or *uniformly learns* $u_i(\cdot)$ within an error of ϵ, if

$\forall x \in [0,1],\ |\hat{u}_i(x) - u_i(x)| \leq \epsilon$. Note that the attacker's mixed strategy remains the same when the utility functions corresponding to all targets are increased by the same value. Therefore, we can only hope to learn a "normalized" representation of the utility functions, $\hat{u}_i$, such that for all i and all x, $|\hat{u}_i(x) + c - u_i(x)| \leq \epsilon$ for some c. Technically, this is exactly what we need

to predict the behavior of the attacker. We use this fact in the proof of our main theorem and choose an appropriate normalization that simplifies the presentation of the technical details.

3.3 Theoretical Results

In this section, we present our theoretical results for learning attacker utility functions. We first state our results in terms of linear utility functions and show that it is possible to uniformly learn the utilities up to error ϵ using only 3 randomized strategies, with $\text{poly}(n, \frac{1}{\epsilon})$ samples for each, under mild conditions. We view these results as practically significant.

In Section 3.3.2, we extend the results to polynomials of degree d to represent a larger class of utility functions. We show (in Section 3.3.3) that this allows us to learn the even more expressive class of Lipschitz utility functions. The extension to high-degree polynomials and Lipschitz utilities requires more restrictive conditions, and hence it is of greater theoretical than practical interest.

Finally, in Section 3.3.4, we show that accurately learning the attacker's utility function allows us to predict the distribution of attacker responses to any defender strategy, and, therefore, to pinpoint a near-optimal strategy.

3.3.1 Linear Utility Functions

Assume that the utility functions are linear and denoted by $u_i(x) = w_i x + c_i$. As discussed in Section 3.2, we can normalize the utilities; That is, without loss of generality $c_n = 0$. Our main result is the following theorem.

Theorem 3.3.1. *Suppose the functions $u_1(\cdot), \ldots, u_n(\cdot)$ are linear. Consider any 3 strategies, $\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathcal{P}$, such that for any $i < n$, $|(p_i - q_i)(p_n - r_n) - (p_n - q_n)(p_i - r_i)| \geq \lambda$, and for any two different strategies $\mathbf{x}, \mathbf{y} \in \{\mathbf{p}, \mathbf{q}, \mathbf{r}\}$, we have $|x_i - y_i| \geq \nu$. If we have access to $m = \Omega(\frac{1}{\rho}(\frac{1}{\epsilon\nu\lambda})^2 \log(\frac{n}{\delta}))$ samples of each of these strategies, then with probability $1 - \delta$, we can uniformly learn each $u_i(\cdot)$ within error ϵ.*

We view the assumptions as being mild. Indeed, intuitively ν depends on how different the strategies are from each other—a very small value means that they are almost identical on some coordinates. The lower bound of λ is less intuitive, but again, it would not be very small unless there is a very specific relation between the strategies. As a sanity check, if the three strategies were chosen uniformly at random from the simplex, both values would be at least $1/\text{poly}(n)$.

To gain some intuition before proceeding with the proof, note that in the quantal best-response model, for each strategy $\mathbf{p}$, the ratio between the attack probabilities of two targets i and n follows the relation

$$u_i(p_i) = \ln\left(\frac{D^{\mathbf{p}}(i)}{D^{\mathbf{p}}(n)}\right) + u_n(p_n). \tag{3.2}$$

Therefore, each strategy induces $n - 1$ linear equations that can be used to solve for the coefficients of u_i. However, we can only obtain an *estimate* $\hat{D}^{\mathbf{p}}(i)$ of the probability that target i is attacked under a strategy $\mathbf{p}$, based on the given samples. So, the inaccuracy in our estimates of

$\ln(\hat{D}^{\mathbf{p}}(i)/\hat{D}^{\mathbf{p}}(n))$ leads to inaccuracy in the estimated polynomial $\hat{u}_i$. For sufficiently accurate estimates $\hat{D}^{\mathbf{p}}(i)$, we show that the value of u_i differs from the true value by at most ϵ.

Let us first analyze the rate of convergence of $\hat{D}^{\mathbf{p}}(i)$ to $D^{\mathbf{p}}(i)$ as the number of observations of strategy $\mathbf{p}$ increases.

Lemma 3.3.2. *Given $\mathbf{p} \in \mathcal{P}$, let $\hat{D}^{\mathbf{p}}(i)$ be the empirical distribution of attacks based on $m = \Omega(\frac{1}{\rho\epsilon^2}\log(\frac{n}{\delta}))$ samples. With probability $1-\delta$, for all $i \in \mathcal{N}$, $\frac{1}{1+\epsilon} \leq \hat{D}^{\mathbf{p}}(i)/D^{\mathbf{p}}(i) \leq 1+\epsilon$.*

Proof. Given $\mathbf{p} \in \mathcal{P}$ and $i \in \mathcal{N}$, let $X_1, \ldots, X_m$ be Bernoulli random variables, whose value is 1 if and only if target i is attacked in sample j, under strategy $\mathbf{p}$. These are i.i.d. random variables with expectation $D^{\mathbf{p}}(i)$. Furthermore, $\hat{D}^{\mathbf{p}}(i) = \frac{1}{m}\sum_j X_j$. Therefore, using the Chernoff bound, we have

$$\Pr\left[\frac{1}{1+\epsilon} \leq \frac{\hat{D}^{\mathbf{p}}(i)}{D^{\mathbf{p}}(i)} \leq 1+\epsilon\right] \geq 1 - 2e^{-mD^{\mathbf{p}}(i)\epsilon^2/4}.$$

Since $D^{\mathbf{p}}(i) > \rho$, when $m = \Omega(\frac{1}{\rho\epsilon^2}\log(\frac{n}{\delta}))$, with probability $1-\frac{\delta}{n}$, $\frac{1}{1+\epsilon} \leq \hat{D}^{\mathbf{p}}(i)/D^{\mathbf{p}}(i) \leq 1+\epsilon$. Taking the union bound over all $i \in \mathcal{N}$, with probability $1-\delta$, for all $i \in \mathcal{N}$, $\frac{1}{1+\epsilon} \leq \hat{D}^{\mathbf{p}}(i)/D^{\mathbf{p}}(i) \leq 1+\epsilon$. □

Proof of Theorem 3.3.1. By Equation 3.2 and using our assumption that $c_n = 0$, for all $i \in \mathcal{N}$, $w_i p_i + c_i = \ln\frac{D^{\mathbf{p}}(i)}{D^{\mathbf{p}}(n)} + w_n p_n$. Using the same equation for $\mathbf{q}$ and eliminating c_i, we have

$$w_i(p_i - q_i) = \ln\frac{D^{\mathbf{p}}(i)}{D^{\mathbf{p}}(n)} - \ln\frac{D^{\mathbf{q}}(i)}{D^{\mathbf{q}}(n)} + w_n(p_n - q_n).$$

Repeating the above for $\mathbf{p}$ and $\mathbf{r}$ and solving for w_n, we have

$$w_n = \frac{(p_i - r_i)\ln\frac{D^{\mathbf{p}}(i)D^{\mathbf{q}}(n)}{D^{\mathbf{q}}(i)D^{\mathbf{p}}(n)} - (p_i - q_i)\ln\frac{D^{\mathbf{p}}(i)D^{\mathbf{r}}(n)}{D^{\mathbf{r}}(i)D^{\mathbf{p}}(n)}}{(p_i - q_i)(p_n - r_n) - (p_n - q_n)(p_i - r_i)}. \tag{3.3}$$

Furthermore, for all $i < n$,

$$w_i = \frac{\ln\frac{D^{\mathbf{p}}(i)}{D^{\mathbf{p}}(n)} - \ln\frac{D^{\mathbf{q}}(i)}{D^{\mathbf{q}}(n)} + w_n(p_n - q_n)}{p_i - q_i} \tag{3.4}$$

and

$$c_i = \ln\frac{D^{\mathbf{p}}(i)}{D^{\mathbf{p}}(n)} + w_n p_n - w_i p_i \tag{3.5}$$

Let $\hat{w}_i$ and $\hat{c}_i$ be defined similarly to w_i and c_i but in terms of the estimates $\hat{D}^{\mathbf{p}}(i)$. By Lemma 3.3.2, for strategy $\mathbf{p}$ (and similarly $\mathbf{q}$ and $\mathbf{r}$) and any i, we have $\frac{1}{1+\epsilon'} \leq \frac{D^{\mathbf{p}}(i)}{\hat{D}^{\mathbf{p}}(i)} \leq 1+\epsilon'$

for $\epsilon' = \epsilon\lambda\nu/128$. Therefore, we have

$$\begin{aligned}
|w_n - \hat{w}_n| &= \frac{\left|(p_i - r_i)\ln\left(\frac{D^{\mathbf{p}}(i)D^{\mathbf{q}}(n)\hat{D}^{\mathbf{q}}(i)\hat{D}^{\mathbf{p}}(n)}{\hat{D}^{\mathbf{p}}(i)\hat{D}^{\mathbf{q}}(n)D^{\mathbf{q}}(i)D^{\mathbf{p}}(n)}\right) - (p_i - q_i)\ln\left(\frac{D^{\mathbf{p}}(i)D^{\mathbf{r}}(n)\hat{D}^{\mathbf{r}}(i)\hat{D}^{\mathbf{p}}(n)}{\hat{D}^{\mathbf{p}}(i)\hat{D}^{\mathbf{r}}(n)D^{\mathbf{r}}(i)D^{\mathbf{p}}(n)}\right)\right|}{|(p_i - q_i)(p_n - r_n) - (p_n - q_n)(p_i - r_i)|} \\
&\leq \frac{|p_i - r_i|\ln(1+\epsilon')^4 + |p_i - q_i|\ln(1+\epsilon')^4}{|(p_i - q_i)(p_n - r_n) - (p_n - q_n)(p_i - r_i)|} \\
&\leq 8\frac{\epsilon'}{\lambda} \\
&\leq \epsilon/16,
\end{aligned}$$

where the third transition follows from the well-known fact that $\ln(1+x) \leq x$ for all $x \in \mathbb{R}$. Similarly, for $i < n$, we have

$$\begin{aligned}
|w_i - \hat{w}_i| &= \frac{\left|\ln\left(\frac{D^{\mathbf{p}}(i)\hat{D}^{\mathbf{p}}(n)}{\hat{D}^{\mathbf{p}}(i)D^{\mathbf{p}}(n)}\right) - \ln\left(\frac{D^{\mathbf{q}}(i)\hat{D}^{\mathbf{q}}(n)}{\hat{D}^{\mathbf{q}}(i)D^{\mathbf{q}}(n)}\right) + (w_n - \hat{w}_n)(p_n - q_n)\right|}{|p_i - q_i|} \\
&\leq \frac{1}{\nu}(4\epsilon' + \epsilon/16) \leq \epsilon/8.
\end{aligned}$$

And,

$$|c_i - c_i| = \left|\ln\frac{D^{\mathbf{p}}(i)\hat{D}^{\mathbf{p}}(n)}{\hat{D}^{\mathbf{p}}(i)D^{\mathbf{p}}(n)} + (w_n - \hat{w}_n)p_n - (w_i - \hat{w}_i)p_i\right| \leq 2\epsilon' + \epsilon/4 \leq \epsilon/2.$$

Therefore, for any i and any $x \in [0,1]$, $|u_i(x) - \hat{u}_i(x)| \leq \epsilon$. □

3.3.2 Polynomial Utility Functions

On the way to learning Lipschitz utilities, we next assume that the utility function is a polynomial of degree at most d (linear functions are the special case of $d = 1$). We show that it is possible to learn these the utility functions using $O(d)$ strategies.

Theorem 3.3.3. *Suppose the functions $u_1(\cdot), \ldots, u_n(\cdot)$ are polynomials of degree at most d. Consider any $2d+1$ strategies, $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(d)}$, $\mathbf{q}^{(d+1)} = \mathbf{p}^{(1)}$, $\ldots, \mathbf{p}^{(d+1)}$, such that for all $k, k', k \neq k'$, $q_1^{(k)} = q_1^{(k')}$, $p_n^{(k)} = p_n^{(k')}$, $|q_n^{(k)} - q_n^{(k')}| \geq \nu$, and for all $i < n$, $|p_i^{(k)} - p_i^{(k')}| \geq \nu$. If we have access to $m = \Omega(\frac{1}{\rho}(\frac{d}{\epsilon\nu^d})^2\log(\frac{n}{\delta}))$ samples of each of these strategies, then with probability $1-\delta$, we can uniformly learn each $u_i(\cdot)$ within error ϵ.*

It is important to emphasize that, unlike Theorem 3.3.1, one would not expect historical data to satisfy the conditions of Theorem 3.3.3, because it requires different strategies to cover some targets with the exact same probability. It is therefore mostly useful in a setting where we have control over which strategies are played. Strictly speaking, these more stringent conditions are not necessary for learning polynomials, but we enforce them to obtain a solution that is stable against inaccurate observations. Also note that the $d = 1$ case of Theorem 3.3.3 is weaker and

less practicable than Theorem 3.3.1, because the latter theorem uses tailor-made arguments that explicitly leverage the structure of linear functions.

In a nutshell, the theorem's proof relies on polynomial interpolation. Specifically, consider the relationship between the utility functions of different targets shown in Equation (3.2). We assume that all the strategies $\mathbf{p}^{(i)}$ have the same coverage probability p_n on target n; since subtracting a fixed constant from all utility functions leaves the distribution of attacks unchanged, we can subtract $u_n(p_n)$ and assume without loss of generality that

$$\forall i < n, \quad u_i(p_i) = \ln\left(\frac{D^{\mathbf{p}}(i)}{D^{\mathbf{p}}(n)}\right). \tag{3.6}$$

Because u_i is a polynomial of degree d, it can be found by solving for the *unique* degree d polynomial that matches the values of u_i at $d+1$ points. To learn u_n, we can then use the same approach with the exception of using the utility function for targets $1, \ldots, n-1$ in Equation (3.2) to get the value of $u_n(\cdot)$ on $d+1$ points. As before, we do not have access to the *exact* values of $D^{\mathbf{p}}(i)$, so we use the estimated values $\hat{D}^{\mathbf{p}}(i)$ in these equations.

The next well-known lemma states the necessary and sufficient conditions for existence of a unique degree d polynomial that fits a collection of $d+1$ points [128].

Lemma 3.3.4. *For any values $y_1, \ldots, y_d$ and $x_1, \ldots, x_d$ such that $x_i \neq x_j$ for all $i \neq j$, there is a unique polynomial $f : \mathbb{R} \to \mathbb{R}$ of degree d, such that for all i, $f(x_i) = y_i$. Furthermore, this polynomial can be expressed as*

$$f(x) = \sum_{k=1}^{d+1} y_k \prod_{k':k' \neq k} \frac{x - x_{k'}}{x_k - x_{k'}}. \tag{3.7}$$

Proof of Theorem 3.3.3. Let $\hat{y}_i^{(k)} = \ln(\hat{D}^{\mathbf{p}^{(k)}}(i)/\hat{D}^{\mathbf{p}^{(k)}}(n))$ for all $i < n$. We have assumed that $p_i^{(k)} \neq p_i^{(k')}$ for any $k \neq k'$, so the conditions of Lemma 3.3.4 hold with respect to the pairs $\left(p_i^{(k)}, \hat{y}_i^{(k)}\right)$. Let $\hat{u}_i$ be the unique polynomial described by Equation (3.7), i.e.,

$$\hat{u}_i(x) = \sum_{k=1}^{d+1} \hat{y}_i^{(k)} \prod_{k':k' \neq k} \frac{x - p_i^{(k')}}{p_i^{(k)} - p_i^{(k')}}.$$

Similarly, for the values $y_i^{(k)} = \ln(D^{\mathbf{p}^{(k)}}(i)/D^{\mathbf{p}^{(k)}}(n))$, by Lemma 3.3.4 and Equation (3.6), $u_i(x)$ can be expressed by

$$u_i(x) = \sum_{k=1}^{d+1} y_i^{(k)} \prod_{k':k' \neq k} \frac{x - p_i^{(k')}}{p_i^{(k)} - p_i^{(k')}}.$$

Let ϵ' be such that $\epsilon = 4\epsilon'(d+1)/\nu^d$. By Lemma 3.3.2 for strategy $\mathbf{p}^{(k)}$ and any i, we have $\frac{1}{1+\epsilon'} \leq \frac{D^{\mathbf{p}^{(k)}}(i)}{\hat{D}^{\mathbf{p}^{(k)}}(i)} \leq 1 + \epsilon'$. Using the fact that $\ln(1+x) \leq x$ for all $x \in \mathbb{R}$, with probability $1 - \delta$,

$$|\hat{y}_i^{(k)} - y_i^{(k)}| = \left|\ln \frac{\hat{D}^{\mathbf{p}^{(k)}}(i)}{D^{\mathbf{p}^{(k)}}(i)} - \ln \frac{\hat{D}^{\mathbf{p}^{(1)}(n)}}{D^{\mathbf{p}^{(1)}(n)}}\right| \leq 2\epsilon'.$$

Therefore, for all x and all $i < n$,

$$|\hat{u}_i(x) - u_i(x)| = \left| \sum_{k=1}^{d+1} (\hat{y}_i^{(k)} - y_i^{(k)}) \prod_{k' \neq k} \frac{x - p_i^{(k')}}{p_i^{(k)} - p_i^{(k')}} \right|$$
$$\leq 2\epsilon' \frac{d+1}{\nu^d} \leq \epsilon/2.$$

Similarly, by Equation (3.2) for target n and $\mathbf{q}^{(k)}$, we have,

$$u_n(q_n^{(k)}) = \ln \left(\frac{D^{\mathbf{q}^{(k)}}(n)}{D^{\mathbf{q}^{(k)}}(1)} \right) + u_1(q_1^{(k)})$$

Since for all k, $q_1^{(k)} = q_1$, using Lemma 3.3.4, u_n can be described by the unique polynomial passing through points $\left(q_n^{(k)}, \ln \frac{D^{\mathbf{q}^{(k)}}(n)}{D^{\mathbf{q}^{(k)}}(1)} \right)$ translated by the value $u^{(1)}(q_1)$. Similarly, let $\hat{u}^{(1)}$ be defined by the unique polynomial passing through points $\left(q_n^{(k)}, \ln \frac{\hat{D}^{\mathbf{q}^{(k)}}(n)}{\hat{D}^{\mathbf{q}^{(k)}}(1)} \right)$ translated by the value $\hat{u}_1(q_1)$, then

$$|\hat{u}_n(x) - u_n(x)| \leq |\hat{u}_1(q_1) - u_1(q_1)| + \left| \sum_{k=1}^{d+1} \left(\ln \frac{\hat{D}^{\mathbf{q}^{(k)}}(n)}{\hat{D}^{\mathbf{q}^{(k)}}(1)} - \ln \frac{D^{\mathbf{q}^{(k)}}(n)}{D^{\mathbf{q}^{(k)}}(1)} \right) \prod_{k':k' \neq k} \frac{x - q_n^{(k')}}{q_n^{(k)} - q_n^{(k')}} \right|$$
$$\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

This completes our proof. □

3.3.3 Lipschitz Utilities

We now leverage the results of Section 3.3.2 to learn any utility function that is continuous and L-Lipschitz, i.e., for all i and values x and y, $|u_i(x) - u_i(y)| \leq L|x - y|$. We argue that such utility functions can be uniformly learned up to error ϵ, using $O(\frac{L}{\epsilon})$ strategies.

To see this, we first state a result that shows that all L-Lipschitz functions can be uniformly approximated within error ϵ using polynomials of degree $O(\frac{L}{\epsilon})$ [121].

Lemma 3.3.5. *Let $\mathcal{F}_m$ be a family of degree m polynomials defined over $[-1, 1]$, and let $\mathcal{F}$ be the set of all L-Lipschitz continuous functions over $[-1, 1]$. Then, for all $f \in \mathcal{F}$,*

$$\inf_{g \in \mathcal{F}_m} \sup_x |f(x) - g(x)| \leq \frac{6L}{m}.$$

Therefore, for any L-Lipschitz function $u_i(x)$, there is a polynomial of degree $m = 12L/\epsilon$ that uniformly approximates $u_i(x)$ within error of $\epsilon/2$. By applying Theorem 3.3.3 to learn polynomials of degree $12L/\epsilon$, we can learn all the utility functions using $O(L/\epsilon)$ strategies.

Corollary 3.3.6. *Suppose the functions $u_1(\cdot), \ldots, u_n(\cdot)$ are L-Lipschitz. For $d = 12L/\epsilon$, consider any $2d+1$ strategies, $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(d)}$, $\mathbf{q}^{(d+1)} = \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(d+1)}$, such that for all $k, k', k \neq k'$, $q_1^{(k)} = q_1^{(k')}$, $p_n^{(k)} = p_n^{(k')}$, $|q_n^{(k)} - q_n^{(k')}| \geq \nu$, and for all $i < n$, $|p_i^{(k)} - p_i^{(k')}| \geq \nu$. If we have access to $m = \Omega(\frac{L^2}{\rho\epsilon^4\nu^{24L/\epsilon}} \log(\frac{n}{\delta}))$ samples of each of these strategies, then with probability $1 - \delta$, we can uniformly learn each $u_i(\cdot)$ within error ϵ.*

3.3.4 Learning the Optimal Strategy

So far, we have focused on the problem of uniformly learning the utility function of the attacker. We now show that an accurate estimate of this utility function allows us to pinpoint an almost optimal strategy for the defender.

Let the utility function of the defender on target $i \in \mathcal{N}$ be denoted by $v_i : [0, 1] \to [-1, 1]$. Given a coverage probability vector $\mathbf{p} \in \mathcal{P}$, the utility the defender receives when target i is attacked is $v_i(p_i)$. The overall expected utility of the defender is

$$\mathcal{V}(\mathbf{p}) = \sum_{i \in \mathcal{N}} D^{\mathbf{p}}(i) v_i(p_i).$$

Let $\hat{u}_i$ be the learned attacker utility functions, and $\bar{D}^{\mathbf{p}}(i)$ be the predicted attack probability on target i under strategy $\mathbf{p}$, according to the utilities $\hat{u}_i$, i.e.,

$$\bar{D}^{\mathbf{p}}(i) = \frac{e^{\hat{u}_i(p_i)}}{\sum_{j \in \mathcal{N}} e^{\hat{u}_j(p_j)}}.$$

Let $\bar{\mathcal{V}}(\mathbf{p})$ be the predicted expected utility of the defender based on the learned attacker utilities $\bar{D}^{\mathbf{p}}(i)$, that is, $\bar{\mathcal{V}}(\mathbf{p}) = \sum_{i \in \mathcal{N}} \bar{D}^{\mathbf{p}}(i) v_i(p_i)$. We claim that when the attacker utilities are uniformly learned within error ϵ, then $\bar{\mathcal{V}}$ estimates $\mathcal{V}$ with error at most 8ϵ. At a high level, this is established by showing that one can predict the attack distribution using the learned attacker utilities. Furthermore, optimizing the defender's strategy against the approximate attack distributions leads to an approximately optimal strategy for the defender.

Theorem 3.3.7. *Assume for all $\mathbf{p}$ and any $i \in \mathcal{N}$, $|\hat{u}_i(p_i) - u_i(p_i)| \leq \epsilon \leq 1/4$. Then, for all $\mathbf{p}$, $|\bar{\mathcal{V}}(\mathbf{p}) - \mathcal{V}(\mathbf{p})| \leq 4\epsilon$. Furthermore, let $\mathbf{p}' = \arg\max_{\mathbf{p}} \bar{\mathcal{V}}(\mathbf{p})$ be the predicted optimal strategy, then $\max_{\mathbf{p}} \mathcal{V}(\mathbf{p}) - \mathcal{V}(\mathbf{p}') \leq 8\epsilon$.*

Proof. First, we show that the predicted attack distribution is close to the real attack distribution

for any strategy $\mathbf{p}$. We have,

$$\begin{aligned}
\left|\ln\left(\frac{\bar{D}^{\mathbf{p}}(i)}{D^{\mathbf{p}}(i)}\right)\right| &= \left|\ln\left(\frac{e^{\hat{u}_i(p_i)}}{e^{u_i(p_i)}}\right) - \ln\left(\frac{\sum_{j\in\mathcal{N}} e^{\hat{u}_j(p_j)}}{\sum_{j\in\mathcal{N}} e^{u_j(p_j)}}\right)\right| \\
&= \left|\hat{u}_i(p_i) - u_i(p_i) - \ln\left(\frac{\sum_{j\in\mathcal{N}} e^{\hat{u}_j(p_j)}}{\sum_{j\in\mathcal{N}} e^{u_j(p_j)}}\right)\right| \\
&\leq \epsilon + \left|\ln\left(\frac{\sum_{j\in\mathcal{N}} e^{u_j(p_j)}\ e^{\hat{u}_j(p_j)-u_j(p_j)}}{\sum_{j\in\mathcal{N}} e^{u_j(p_j)}}\right)\right| \\
&\leq \epsilon + \max_j \left|\ln(e^{\hat{u}_j(p_j)-u_j(p_j)})\right| \\
&\leq 2\epsilon.
\end{aligned}$$

Using the well-known inequalities $1-x \leq e^{-x}$, $e^{x-x^2/2} < 1+x$, and $2\epsilon \leq 4\epsilon - 8\epsilon^2$ for $\epsilon \leq 1/4$, we have

$$(1-2\epsilon) \leq e^{-2\epsilon} \leq \frac{\bar{D}^{\mathbf{p}}(i)}{D^{\mathbf{p}}(i)} \leq e^{2\epsilon} \leq e^{4\epsilon-8\epsilon^2} \leq (1+4\epsilon).$$

Then,

$$\begin{aligned}
|\bar{\mathcal{V}}(\mathbf{p}) - \mathcal{V}(\mathbf{p})| &\leq \sum_{i\in\mathcal{N}} v_i(p_i)\ |\bar{D}^{\mathbf{p}}(i) - D^{\mathbf{p}}(i)| \\
&= \sum_{i\in\mathcal{N}} v_i(p_i)\ \left|\frac{\bar{D}^{\mathbf{p}}(i)}{D^{\mathbf{p}}(i)} - 1\right| D^{\mathbf{p}}(i) \\
&\leq 4\epsilon \sum_{i\in\mathcal{N}} v_i(p_i) D^{\mathbf{p}}(i) \\
&\leq 4\epsilon\ \max_i v_i(p_i) \left(\sum_{i\in\mathcal{N}} D^{\mathbf{p}}(i)\right) \\
&\leq 4\epsilon.
\end{aligned}$$

Let $\mathbf{p}^* = \arg\max_{\mathbf{p}} \mathcal{V}(\mathbf{p})$ be the true defender's optimal strategy. Then, $\mathcal{V}(\mathbf{p}^*) \leq \bar{\mathcal{V}}(\mathbf{p}^*) + 4\epsilon \leq \bar{\mathcal{V}}(\mathbf{p}') + 4\epsilon \leq \mathcal{V}(\mathbf{p}') + 8\epsilon$. This completes the proof. □

3.4 Discussion and Open Problems

In this chapter, we considered learning in Stackelberg security games when the attacker behavior deviates from the best-response function. In particular, we studied the Subjective Utility Quantal Response models — a commonly-used model from behavioral game theory — where rather than attacking the most lucrative target $i^* = \arg\max u_i(p_i)$, the attacker attacks any target i with probability proportional to $\exp(u_i(p_i))$. Motivated by the need for algorithms that withstand unexpected attacker behavior, a natural research direction in Stackelberg security games is to consider deviations from the best-response model that are even less stylized.

In machine learning, such deviations from the "correct" behavior are collectively referred to as *noise*. For example, in the context of classification, a no noise (realizable) setting refers to the case where the label of each instance x is $y = f^*(x)$ for a fixed (but unknown) f^* in a predetermined class of functions. A variety of noise models have been studied in this context; from more stylized noise models where the probability of having a correct label, $\Pr[y = f^*(x) \mid x] = \eta$, is known and uniform over all x, such as the *random classification noise model*; to highly asymmetric noise models where the probability of having a correct label is bounded but not uniform, i.e., $\Pr[y = f^*(x) \mid x] = \eta(x) \geq \eta$, called the *bounded* noise model. There has been significant push towards addressing highly unstructured noise models, such as the results presented in Chapters 8 and 9. This has led to learning algorithms that are robust to a wider range of deviation from a prescribed behavior.

Inspired by the shift from stylized noise models towards highly unstructured ones, one possible relaxation of the Subjective Utility Quantal Response model is as follows. Consider a setting where the best response $i^* = \arg\max u_i(p_i)$ is attacked with probability at least $\exp(u_i(p_i))$. How many short-term or long-term observations are required to learn the optimal defender strategy? We leave this question unanswered in this thesis, but we discuss learning with an analogous classification noise in Chapters 8 and 9.

Chapter 4

Online Learning in Multi-attacker Stackelberg Games

4.1 Introduction

In this chapter, we continue our study of Stackelberg security games where we face uncertain attacker utility and behavior. As we have seen in Chapters 2 and 3 and related works [56, 190, 203] one way to alleviate uncertainty about the attacker behavior is to learn about it through repeated interaction with the attacker: at each round, the defender commits to a mixed strategy and observes the attacker's best response. But this line of work is restricted to repeated interaction with a single attacker type [56, 203], or simple variations thereof [190]. Either way, the defender faces the exact same situation in each round—a convenient fact that allows the learning process to ultimately converge to an (almost) optimal strategy, which is then played until the end of time.

In this chapter, we deal with uncertainty about attackers by adopting a fundamentally different approach, which makes a novel connection to the extensive literature on *online learning*. Similarly to previous work [56, 190, 203], we study a repeated Stackelberg game; but in our setting the attackers are not all the same—in fact, they are chosen *adversarially* from some known set of types Θ. That is, at each round the defender commits to a mixed strategy based on the history of play so far, and an adversarially chosen attacker from Θ best-responds to that strategy.

Even in the face of this type of uncertainty, we would like to compete with the best fixed mixed strategy *in hindsight*, that is, the mixed strategy that yields the highest total payoff when played against each attacker in the sequence generated by the adversary. The *regret* associated with an online learning algorithm (which recommends to the defender a mixed strategy at each step) is simply the difference between the utility of the best-in-hindsight fixed strategy and the expected utility of the algorithm in the online setting. Our goal is to

> *... design online learning algorithms whose regret is sublinear in the number of time steps, and polynomial in the parameters of the game.*

Such an algorithm—whose average regret goes to zero as the number of time steps goes to infinity—is known as a *no-regret algorithm*. While there has been substantial work on no-regret

learning, what makes our situation different is that our goal is to compete with the best mixed strategy in hindsight against the sequence of attackers that arrived, *not* the sequence of targets they attacked.

4.1.1 Overview of Our Results

We provide two algorithmic results that apply to two different models of feedback. In the *full information* model (Section 4.5), the defender plays a mixed strategy, and observes the type of attacker that responds. This means that the algorithm can infer the attacker's best response to *any* mixed strategy, not just the one that was played. We design an algorithm whose regret is $O\left(\sqrt{Tn^2k\log(nk)}\right)$, where T is the number of time steps, n is the number of targets that can be attacked, and k is the number of attacker types.

In the second model—the *partial information* model (Section 4.6)—the defender only observes which target was attacked at each round. Our main technical result is the design and analysis of a no-regret algorithm in the partial information model whose regret is bounded by $O\left(T^{2/3}nk\log^{1/3}(nk)\right)$.

For both results we assume that the attackers are selected (adversarially) from a set of k *known* types. It is natural to ask whether no-regret algorithms exist when there are no restrictions on the types of attackers. In Section 4.7, we answer this question in the negative, thereby justifying the dependence of our bounds on k.

Let us make two brief remarks regarding central issues that are discussed at length in Section 4.8. First, throughout this chapter we view *information*, rather than *computation*, as the main bottleneck, and therefore aim to minimize regret without worrying (for now) about computational complexity. Second, our exposition focuses on Stackelberg *Security* Games, but our framework and results apply to Stackelberg games more generally.

4.1.2 Related work

This chapter presents the first treatment of Stackelberg Security games in the online learning setting.

Our work in this chapter is closely related to work on online learning. For the full information feedback case, the seminal work of Littlestone and Warmuth [196] achieves a regret bound of $O\left(\sqrt{T\log N}\right)$ for N strategies. Kalai and Vempala [169] show that when the set of strategies is a subset of $\mathbb{R}^d$ and the loss function is linear, this bound can be improved to $O(\sqrt{TD})$, where D is the ℓ_1-diameter of the strategy set. This bound is applicable when there are infinitely many strategies. In our work, the space of mixed strategies can be translated to a n-dimensional space, where n is the number of targets. However, our loss function depends on the best response of the attackers, hence is not linear.

For the partial feedback case (also known as the *bandit* setting), Auer et al. [20] introduce an algorithm that achieves regret $O(\sqrt{TN\log N})$ for N strategies. Awerbuch and Kleinberg [27] extend the work of Kalai and Vempala [169] to the partial information feedback setting for online routing problems, where the feedback is in the form of the end-to-end delay of a chosen

source-to-sink path. Their approach can accommodate exponentially or infinitely large strategy spaces, but again requires a linear loss function.

4.2 Preliminaries

We consider a repeated Stackelberg security game between a defender (the leader) and a sequence of attackers (the followers). At each step of this repeated game, the interactions between the defender and the attacker induce a Stackelberg security game, where the defender commits to a randomized allocation of his security resources to defend potential targets, and the attacker, in turn, observes this randomized allocation and attacks the target with the best expected payoff. The defender and the attacker then receive payoffs. The defender's goal is to maximize his payoff over a period of time, even when the sequence of attackers is unknown.

More precisely, a *repeated stackelberg security game* includes the following components:

- *Time horizon* T: the number of rounds.
- Set of *targets* $\mathcal{N} = \{1, \ldots, n\}$.
- A defender with:
  - *Resources*: A set of *resources* R.
  - *Schedules:* A collection $\mathcal{D} \subseteq 2^{\mathcal{N}}$ of schedules. Each schedule $D \in \mathcal{D}$ represents a set of targets that can be simultaneously defended by one resource.
  - *Assignment function*: Function $A : R \to 2^{\mathcal{D}}$ indicates the set of all schedules that can be defended by a given resource. An assignment of resources to schedules is *valid* if every resource r is allocated to a schedule in $A(r)$.
  - *Strategy*: A *pure strategy* is a valid assignment of resources to schedules. The set of all pure strategies is determined by $\mathcal{N}, \mathcal{D}, R$, and A and can be represented as follow. Let there be m pure strategies, and let M be a zero-one $n \times m$ matrix, such that the rows represent targets and columns represent pure strategies, with $M_{i,j} = 1$ if and only if target i is covered by some resource in the j^{th} pure strategy.

    A *mixed strategy* is a distribution over the set of pure strategies and is represented by an $m \times 1$ probability vector $\mathbf{s}$, such that for all j, s_j is the probability with which pure strategy j is played. Every mixed strategy induces a *coverage probability* vector $\mathbf{p} \in [0, 1]^n$, where p_i is the probability with which target i is defended under that mixed strategy. The mapping between a mixed strategy, $\mathbf{s}$, and its coverage probability vector, $\mathbf{p}$, is given by $\mathbf{p} = M\mathbf{s}$.
- A set of all attacker types $\Theta = \{\theta_1, \ldots, \theta_k\}$.
  - An adversary selects a *sequence of attackers* $\mathbf{a} = a_1, \ldots, a_T$, such that for all t, $a_t \in \Theta$.
  - Throughout this work we assume that Θ is known to the defender, while $\mathbf{a}$ remains unknown.

- *Utilities*: The defender and attacker both receive payoffs when a target is attacked.
  - For the defender, let $u_d^c(i)$ and $u_d^u(i)$ be the defender's payoffs when target i is attacked and is, respectively, covered or not covered by the defender. Then, under coverage probability $\mathbf{p}$, the expected utility of the defender when target i is attacked is given by $U_d(i, \mathbf{p}) = u_d^c(i)p_i + u_d^u(i)(1 - p_i)$. Note that $U_d(i, \mathbf{p})$ is linear in $\mathbf{p}$. All utilities are normalized such that $u_d^c(i), u_d^u(i) \in [-1, 1]$ for all $i \in \mathcal{N}$.
  - For an attacker of type θ_j, let $u_{\theta_j}^c(i)$ and $u_{\theta_j}^u(i)$ be the attacker's payoffs from attacking target i when the target is, respectively, covered or not covered by the defender. Then under coverage probability $\mathbf{p}$, the expected utility of the attacker from attacking target i is given by $U_{\theta_j}(i, \mathbf{p}) = u_{\theta_j}^c(i)p_i + u_{\theta_j}^u(i)(1 - p_i)$. Note that $U_{\theta_j}(i, \mathbf{p})$ is linear in $\mathbf{p}$.

At step t of the game, the defender chooses a mixed strategy to deploy over his resources. Let $\mathbf{p}_t$ be the coverage probability vector corresponding to the mixed strategy deployed at step t. Attacker a_t observes $\mathbf{p}_t$ and *best-responds* to it by attacking target $b_{a_t}(\mathbf{p}) = \arg\max_{i \in \mathcal{N}} U_{a_t}(i, \mathbf{p})$. When multiple targets have the same payoff to the attacker, each attacker breaks ties in some arbitrary but consistent order.

Note that given a coverage probability vector, the utilities of all players and the attacker's best-response is invariant to the mixed strategy that is used to implement that coverage probability. Therefore, we work directly in the space of valid coverage probability vectors, denoted by $\mathcal{P}$. For ease of exposition, in the remainder of this work we do not distinguish between a mixed strategy and its coverage probability vector.

4.3 Problem Formulation

In this section, we first formulate the question of *how a defender can effectively protect targets in a repeated Stackelberg security game when the sequence of attackers is not known to him.* We then give an overview of our approach.

Consider a repeated Stackelberg security setting with one defender and a sequence of attackers, $\mathbf{a}$, that is selected beforehand by an adversary. When $\mathbf{a}$ is not known to the defender a priori, the defender has to adopt an online approach to maximize his overall payoff during the game. That is, at every time step t the defender plays, possibly at random, a mixed strategy $\mathbf{p}_t$ and receives some "feedback" regarding the attacker at that step. The defender subsequently adjusts his mixed strategy $\mathbf{p}_{t+1}$ for the next time step. The expected payoff of the defender is given by

$$\mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right],$$

where, importantly, the expectation is taken only over internal randomness of the defender's online algorithm.

The feedback the defender receives at each time step plays a major role in the design of the algorithm. In this work, we consider two types of feedback, *full information* and *partial information*. In the full information case, after attacker a_t best-responds to $\mathbf{p}_t$ by attacking a

target, the defender observes a_t, that is, the defender know which attacker type just attacked. In the partial information case, even after the attack occurs the defender only observes the target that was attacked, i.e. $b_{a_t}(\mathbf{p}_t)$. More formally, in the full-information and partial-information settings, the choice of $\mathbf{p}_t$ depends on a_i for all $i \in [t-1]$ or $b_{a_i}(\mathbf{p}_i)$ for all $i \in [t-1]$, respectively. As a sanity check, note that knowing a_i is sufficient to compute $b_{a_i}(\mathbf{p}_i)$, but multiple attacker types may respond by attacking the same target; so feedback in the full information case is indeed strictly more informative.

To examine the performance of our online approach, we compare its payoff to the defender's payoff in a setting where $\mathbf{a}$ is known to the defender. In the event that the sequence of attackers, or merely the frequency of each attacker type in the sequence, is known, the defender can pre-compute a fixed mixed strategy with the best payoff against that sequence of attackers and play it at every step of the game. We refer to this mixed strategy as the *best mixed strategy in hindsight* and denote it by

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}), \mathbf{p}).^{1}$$

Our goal is to design online algorithms for the defender with payoff that is almost as good as the payoff of the best mixed strategy in hindsight. We refer to the difference between the utility of the online algorithm and the utility of the best-in-hindsight mixed strategy as *regret*, i.e.,

$$\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - \mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right].$$

Our results are stated as upper and lower bounds on regret. For upper bounds, we show both in the case of full information feedback (Theorem 4.5.1) and partial information feedback (Theorem 4.6.1) that

$$\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - \mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] \leq o(T) \cdot \mathrm{poly}(n, k).$$

In particular, the average regret goes to zero as T goes to infinity, that is, our algorithms are *no-regret algorithms*. In contrast, we show that even in the full-information case, when Θ is large compared to T, our regret must be linear in T (Theorem 4.7.1).

4.3.1 Methodology

This formulation of our problem, which involves comparison between online and offline decision making, closely matches (by design) the classic online learning framework. To formally introduce this setup, consider a set of actions $\mathcal{M}$, time horizon T, an online learner, and an *adaptive* adversary. For every $t \in [T]$, the learner chooses a distribution $\mathbf{q}_t$ over the actions in $\mathcal{M}$, and

[1] As we explain in Section 4.4, for a general tie breaking rule, it is possible that the optimal strategy $\mathbf{p}^*$ may not be well-defined. However, the value of the optimal strategy is always well-defined in the limit. Slightly abusing notation for ease of exposition, we use $\mathbf{p}^*$ to refer to a strategy that acheives this optimal value in the limit.

then picks a random action based on this distribution, indicated by j_t. The adversary then chooses a loss vector, $\boldsymbol{\ell}_t$, such that for all j, $\boldsymbol{\ell}_t(j) \in [-\kappa, \kappa]$. The adversary is adaptive, in the sense that the choice of $\boldsymbol{\ell}_t$ can depend on the distributions at every step $\mathbf{q}_1, \ldots, \mathbf{q}_t$, and on the realized actions in the previous steps $j_1, \ldots, j_{t-1}$. The online learner then incurs an expected loss of $\mathbf{q}_t \cdot \boldsymbol{\ell}_t$.

Let $L_{alg} = \sum_{t=1}^{T} \mathbf{q}_t \cdot \boldsymbol{\ell}_t$ be the total expected loss of the online learner over time period T for a choice of an algorithm. On the other hand, let $L_{\min} = \min_{j \in \mathcal{M}} \sum_{t=1}^{T} \boldsymbol{\ell}_t(j)$, be the loss of the best fixed action for the sequence $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T$. Define *regret* as $R_{T,\mathcal{M},\kappa} = L_{alg} - L_{min}$.

In our work we leverage a well-known result on no-regret learning, which can be stated as follows (see, e.g., [53]).

Proposition 4.3.1. *There is an algorithm such that*

$$R_{T,\mathcal{M},\kappa} \leq \sqrt{T\kappa \log(|\mathcal{M}|)}$$

In this work, we can use any algorithm that satisfies the above guarantee as a black box. Many algorithms fall into this category, e.g., *Polynomial Weights* [72] and *Follow the Lazy Leader* [169]. In Section 4.8, we discuss the choice of algorithm further. Also note that any utility maximization problem has an equivalent loss minimization formulation. To be consistent with the notation used by the learning community, we adopt the notion of loss minimization when dealing with results pertaining to online learning.

Although our problem is closely related to classic regret minimization, our goal cannot be readily accomplished by applying Proposition 4.3.1. Indeed, each *mixed* strategy in a security game corresponds to one action in Proposition 4.3.1. This creates an infinitely large set of actions, which renders the guarantees given by Proposition 4.3.1 meaningless. Previous work resolves this issue for a subset of problems where the action space is itself a vector space and the loss function has some desirable properties, e.g., linear or convex with some restrictions [66, 169, 274]. However, the loss function used in our work does not have such nice structural properties: the loss function at step t depends on the best response of the attacker, which leads to a loss function that is not linear, convex, or even continuous in the mixed strategy.

4.4 Characteristics of the Offline Optimum

In this section, we examine the characteristics of the best-in-hindsight mixed strategy. Using this characterization, we choose a set of mixed strategies that are representative of the continuous space of all mixed strategies. That is, rather than considering all mixed strategies in $\mathcal{P}$, we show that we can limit the choices of our online algorithm to a subset of mixed strategies without incurring (significant) additional regret.

To this end, we first show that the given attacker types partition the space of mixed strategies into convex regions where the attacker's best response remains fixed.

Definition 4.4.1. *For every target $i \in \mathcal{N}$ and attacker type $\theta_j \in \Theta$, let $\mathcal{P}_i^j$ indicate the set of all valid coverage probabilities where an attacker of type θ_j attacks target i, i.e.,*

$$\mathcal{P}_i^j = \{\mathbf{p} \in \mathcal{P} \mid b_{\theta_j}(\mathbf{p}) = i\}.$$

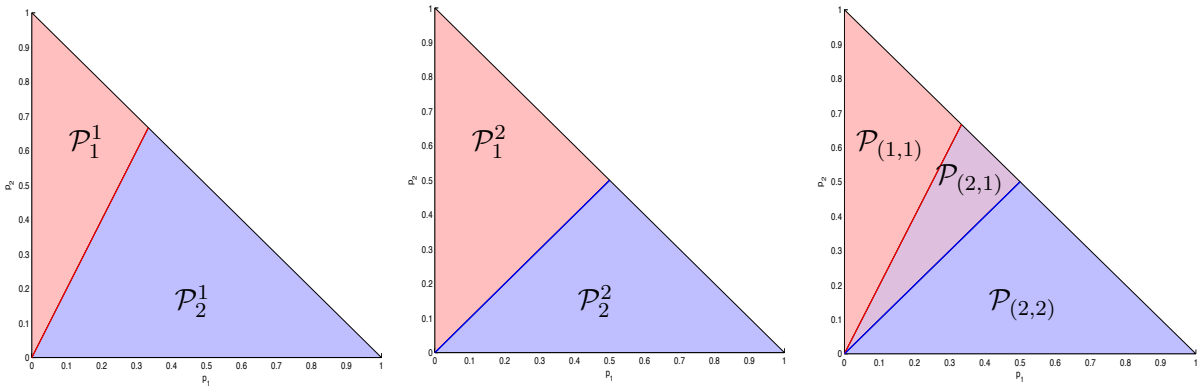


Figure 4.1: Best-response regions. The first two figures define $\mathcal{P}_i^j$ in a game where one resource can cover one of two targets, and two attacker types. The third figure illustrates $\mathcal{P}_\sigma$ for the intersection of the best-response regions of the two attackers.

It is known that for all i and j, $\mathcal{P}_i^j$ is a convex polytope [56].

Definition 4.4.2. *For a given function $\sigma : \Theta \to \mathcal{N}$, let $\mathcal{P}_\sigma$ indicate the set of all valid coverage probability vectors such that for all $\theta_j \in \Theta$, θ_j attacks $\sigma(j)$. In other words, $\mathcal{P}_\sigma = \bigcap_{j \in \Theta} \mathcal{P}^j_{\sigma(j)}$.*

Note that $\mathcal{P}_\sigma$ is an intersection of finitely many convex polytopes, so it is itself a convex polytope. Let Σ indicate the set of all σ for which $\mathcal{P}_\sigma$ is a non-empty region. Figure 4.1 illustrates these regions.

The next lemma essentially shows that the optimal strategy in hindsight is an extreme point of one of the convex polytopes defined above. There is one subtlety, though: due to tie breaking, for some σ, $\mathcal{P}_\sigma$ is not necessarily closed. Hence, some of the extreme points of the closure of $\mathcal{P}_\sigma$ are not within that region. To circumvent this issue, instead we prove that the optimal strategy in hindsight is *approximately* an extreme point of one of the regions. That is, for a given $\epsilon > 0$, we define $\mathcal{E}$ to be a set of mixed strategies as follow: for all σ and any $\mathbf{p}$ that is an extreme point of the closure of $\mathcal{P}_\sigma$, if $\mathbf{p} \in \mathcal{P}_\sigma$, then $\mathbf{p} \in \mathcal{E}$, otherwise there exists $\mathbf{p}' \in \mathcal{E}$ such that $\mathbf{p}' \in \mathcal{P}_\sigma$ and $||\mathbf{p} - \mathbf{p}'||_1 \leq \epsilon$.

Lemma 4.4.3. *Let $\mathcal{E}$ be as defined above, then for any sequence of attackers $\mathbf{a}$,*

$$\max_{\mathbf{p} \in \mathcal{E}} \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}), \mathbf{p}) \geq \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - 2\epsilon T.$$

Proof. Recall that $\mathbf{p}^*$ is the optimal strategy in hindsight for a sequence of attackers $\mathbf{a}$. Since the set of regions $\mathcal{P}_\sigma$ for all $\sigma : \Theta \to \mathcal{N}$ partitions the set of all mixed strategies, $\mathbf{p}^* \in \mathcal{P}_\sigma$ for some σ. We will show that there is a point $\mathbf{p}'' \in \mathcal{E}$ that is near optimal for the sequence $\mathbf{a}$.

For any coverage probability $\mathbf{p} \in \mathcal{P}_\sigma$, let $\mathcal{U}_\mathbf{a}(\mathbf{p})$ be the defender's expected payoff for playing $\mathbf{p}$ against sequence $\mathbf{a}$. Then,

$$\mathcal{U}_\mathbf{a}(\mathbf{p}) = \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}), \mathbf{p}) = \sum_{t=1}^{T} U_d(\sigma(a_t), \mathbf{p}) = \sum_{i=1}^{N} U_d(i, \mathbf{p}) \sum_{t=1}^{T} \mathbb{I}(\sigma(a_t) = i),$$

where $\mathbb{I}$ is the indicator function. Note that $\sum_{t=1}^{T} \mathbb{I}(\sigma(a_t) = i)$, which is the total number of times target i is attacked in the sequence $\mathbf{a}$, is constant over $\mathcal{P}_\sigma$. Moreover, by the definition of utilities, $U_d(i, \mathbf{p})$ is a linear function in $\mathbf{p}$. Therefore, $\mathcal{U}_\mathbf{a}(\mathbf{p})$ is a summation of linear functions, and as a result, is itself a linear function in $\mathbf{p}$ over $\mathcal{P}_\sigma$.

Let $\mathcal{P}'_\sigma$ be the closure of $\mathcal{P}_\sigma$. So, $\mathcal{P}'_\sigma$ is a closed convex polytope. Since $\mathcal{U}_\mathbf{a}(\mathbf{p})$ is a linear function over the convex set $\mathcal{P}'_\sigma$, there is an extreme point $\mathbf{p}' \in \mathcal{P}'_\sigma$ such that

$$\sum_{t=1}^{T} U_d(\sigma(a_t), \mathbf{p}') \geq \mathcal{U}_\mathbf{a}(\mathbf{p}^*).$$

Now, let $\mathbf{p}'' \in \mathcal{E} \cap \mathcal{P}_\sigma$ be the point that corresponds to extreme point $\mathbf{p}'$, i.e. $||\mathbf{p}' - \mathbf{p}''||_1 \leq \epsilon$. Since for each target $i \in \mathcal{N}$, $u_d^c(i), u_d^u(i) \in [-1, 1]$ we have

$$U_d(i, p_i'') = u_d^c(i)p_i'' + u_d^u(i)(1 - p_i'') \geq u_d^c(i)(p_i'' + \epsilon) + u_d^u(i)(1 - p_i'' - \epsilon) - 2\epsilon \geq U_d(i, p_i') - 2\epsilon.$$

Hence,

$$\mathcal{U}_\mathbf{a}(\mathbf{p}'') = \sum_{t=1}^{T} U_d(\sigma(a_t), \mathbf{p}'') \geq \sum_{t=1}^{T} U_d(\sigma(a_t), \mathbf{p}') - 2\epsilon T \geq \mathcal{U}_\mathbf{a}(\mathbf{p}^*) - 2\epsilon T.$$

□

The above lemma shows that by only considering strategies in $\mathcal{E}$, our online algorithm will merely incur an additional loss of ϵT. For a small enough choice of ϵ this only adds a lower-order term to our regret analysis, and as a result can effectively be ignored. For the remainder of this chapter, we assume that $\mathcal{E}$ is constructed with $\epsilon \in O(\frac{1}{\sqrt{t}})$. Next, we derive an upper bound on the size of $\mathcal{E}$.

Lemma 4.4.4. *For any repeated security game with n targets and k attacker types, $|\mathcal{E}| \in O((2^n + kn^2)^n n^k)$.*

Proof. Any extreme point of a convex polytope in n dimensions is the intersection of n linearly independent defining half-spaces of that polytope. To compute the number of extreme points, we first compute the number of defining half-spaces. For a given attacker type θ_j, there are $\binom{n}{2}$ half-spaces each separating $\mathcal{P}_i^j$ and $\mathcal{P}_{i'}^j$ for two targets $i \neq i'$. Summing over all attacker types, there are $O(kn^2)$ such half-spaces. Moreover, the region of the valid coverage probabilities is itself the intersection of $O(m + n)$ half-spaces, where m is the number of subsets of targets that are covered in some pure strategy [56]. In the worst case, $m = 2^n$. Therefore, each extreme point is an intersection of n halfspaces chosen from $O(2^n + n + kn^2)$ half-spaces, resulting in at most $\binom{m+n+kn^2}{n} \in O((2^n + kn^2)^n)$ extreme points. Each extreme point can give rise to at most n^k other ϵ-approximate extreme points. Therefore, $|\mathcal{E}| \in O((2^n + kn^2)^n n^k)$. □

4.5 Upper bounds – Full Information

In this section, we establish an upper bound on regret in the full information setting. With the machinery of Section 4.4 in place, the proof follows quite easily from Proposition 4.3.1.

Theorem 4.5.1. *Given a repeated security game with full information feedback, n targets, k attacker types, and time horizon T[2], there is an online algorithm that for any unknown sequence of attackers, $\mathbf{a}$, at time t plays a randomly chosen mixed strategy $\mathbf{p}_t$, and has the property that*

$$\mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] \geq \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - O\left(\sqrt{Tn^2k\log nk}\right),$$

where the expectation is taken over the algorithm's internal randomization.

Before proving the theorem, a comment is in order. The algorithm is playing a distribution over mixed strategies at any stage; isn't that just a mixed strategy? The answer is no: the attacker is best-responding to the mixed strategy drawn from the distribution over mixed strategies chosen by the defender. The best responses of the attacker could be completely different if he responded directly to the mixed strategy induced by this distribution over mixed strategies.

Proof of Theorem 4.5.1. We use any algorithm that satisfies Proposition 4.3.1. Let the set of extreme points $\mathcal{E}$ denote the set of actions to be used in conjunction with this algorithm. At every round, after observing a_t, we compute the loss of all mixed strategies $\mathbf{p} \in \mathcal{E}$ by setting $\ell_t(\mathbf{p}) = -U_d(b_{a_t}(\mathbf{p}), \mathbf{p})$. Note that the problem of maximizing utility is now transformed to minimizing the loss. Using Proposition 4.3.1 and Lemma 4.4.4, we have

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] &\geq \max_{\mathbf{p}\in\mathcal{E}} \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}), \mathbf{p}) - O(\sqrt{T\log|\mathcal{E}|}) \\
&\geq \max_{\mathbf{p}\in\mathcal{E}} \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}), \mathbf{p}) - O\left(\sqrt{T(n\log(2^n + kn^2) + k\log n)}\right) \\
&\geq \max_{\mathbf{p}\in\mathcal{E}} \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}), \mathbf{p}) - O\left(\sqrt{Tn^2k\log nk}\right).
\end{aligned}$$

Using Lemma 4.4.3 with an appropriate choice of $\epsilon \in O(\frac{1}{\sqrt{T}})$, we have

$$\mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] \geq \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - O\left(\sqrt{Tn^2k\log nk}\right).$$

□

4.6 Upper bounds – Partial Information

In this section, we establish an upper bound on regret in the partial information setting. Recall that under partial information feedback, after an attack has occurred, the defender only observes the target that was attacked, not the attacker type that initiated the attack. Our goal is to prove the following theorem.

[2]If T is unknown, we can use the *guess and double* technique, adding to the regret bound an extra constant factor.

Theorem 4.6.1. *Given a repeated security game with partial information feedback, n targets, k attacker types, and time horizon T, there is an online algorithm that for any unknown sequence of attackers, $\mathbf{a}$, at time t plays a randomly chosen mixed strategy $\mathbf{p}_t$, and has the property that*

$$\mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] \geq \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - O\left(T^{2/3}\, nk\, \log^{1/3}(nk)\right),$$

where the expectation is taken over algorithm's internal randomization.

4.6.1 Overview of the Approach

The central idea behind our approach is that any regret-minimization algorithm in the full information setting also works with partial feedback if, instead of observing the loss of every action at a time step, it receives an *unbiased estimator* of the loss of all actions. For time horizon T and range of action loss $[-\kappa, +\kappa]$, let $R_{T,\kappa}$ represent the regret of a full-information algorithm. For any action j and any time step t, let $\hat{\ell}_t(j)$ be an unbiased estimator of the loss of action j at time step t. Run the full-information regret-minimization algorithm with the values of the loss estimators, i.e. $\hat{\ell}_t(j)$, and let $q_t(j)$ be the probability that our full-information algorithm picks action j at time t. Then, the expected loss of this algorithm (denoted $L_{Est.}$) is as follows.

$$\begin{aligned} L_{Est.} &= \sum_{t=1}^{T}\sum_{j\in\mathcal{M}} q_t(j)\ell_t(j) = \sum_{t=1}^{T}\sum_{j\in\mathcal{M}} q_t(j)\,\mathbb{E}[\hat{\ell}_t(j)] = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{j\in\mathcal{M}} q_t(j)\hat{\ell}_t(j)\right] \\ &\leq \mathbb{E}\left[\min_j \sum_{t=1}^{T}\hat{\ell}_t(j) + R_{T,\kappa}\right] \leq \min_j \mathbb{E}\left[\sum_{t=1}^{T}\hat{\ell}_t(j)\right] + R_{T,\kappa} = \min_j \sum_{t=1}^{T}\ell_t(j) + R_{T,\kappa}. \end{aligned} \tag{4.1}$$

This means that, in the partial information setting, we can use a full-information regret-minimization algorithm in combination with a mechanism to estimate the loss of all actions. This is similar to the approach first introduced by Awerbuch and Kleinberg [27]. Informally speaking, we simulate one time step of a full information setting by a window of time, where we mostly use the mixed strategy that is suggested by the full information algorithm (exploitation), except for a few time steps where we invoke a mechanism for getting an estimate for the loss of all mixed strategies in that window of time (exploration). We then pass these estimates to the full-information algorithm to be used for future time steps.

A naïve approach for getting unbiased estimators of the loss of all mixed strategies involves sampling each mixed strategy once at random in a window of time and observing its loss. Note that, at every time step in which we sample a random strategy, we could incur significant regret. That is, the strategy that is being sampled may have significant loss compared to the strategy suggested by the full-information regret minimization algorithm. Therefore, sampling each strategy individually once at random adds regret that is polynomial in the number of mixed strategies sampled, which in our case is *exponential* in the number of targets and types (See Lemma 4.6.2 for more details). Therefore, a more refined mechanism for finding an unbiased

estimator of the loss is needed. That is, the question is: *How can we get an unbiased estimator of the loss of all mixed strategies while sampling only a few of them?*

To answer this question, we first notice that the loss of different mixed strategies is not independent, rather they all depend on the number of times each target is attacked, which in turn depends on the type frequency of attackers. The challenge, then, is to infer an unbiased estimator of the type frequencies, by only observing the best responses (not the types themselves).

As an example, assume that there is a mixed strategy where each attacker type responds differently (See Figure 4.1). Then by observing the best-response, we can infer the type with certainty. In general, such a mixed strategy, where each attacker responds differently, might not exists. This is where insights from *bandit linear optimization* prove useful.

In more detail, in order to estimate the loss of a mixed strategy $\mathbf{p} \in \mathcal{P}_\sigma$, it is sufficient to estimate the total frequency of the set of attacker types that attack target i in region $\mathcal{P}_\sigma$, for any $i \in \mathcal{N}$. This value itself is *a linear function* in $\mathbb{R}^k$. That is, let $\mathbb{I}_{(\sigma=i)}$ be the indicator vector of the set of attackers that attack i in region $\mathcal{P}_\sigma$ and let $\mathbf{f} = (f_1, \ldots, f_k) \in \mathbb{R}^k$ be the vector of frequencies of attacker types. The loss of mixed strategy $\mathbf{p}$ can be determined using values $\mathbf{f} \cdot \mathbb{I}_{(\sigma=i)}$ for all $i \in \mathcal{N}$. Moreover, even though we cannot observe the above inner products directly under partial information feedback, we can create an unbiased estimator of any $\mathbf{f} \cdot \mathbb{I}_{(\sigma=i)}$ by sampling any $\mathbf{p} \in \mathcal{P}_\sigma$ and observing how often target i is attacked in response. Therefore our problem reduces to creating "good" unbiased estimators for the above inner products, which lie in a k-dimensional vector space. This can be achieved by only sampling a set of k strategies (See Lemmas 4.6.4 and 4.6.5)

One subtle obstacle remains, and that is the range of the new loss estimator (where the notion of a "good" estimator comes in). Indeed, as shown in Eq. 4.1, the regret also depends on the range of the loss estimator. To this end, we use the concept of *barycentric spanners* [27] that is also used in bandit linear optimization, to ensure that the range of the loss estimator remains small even after inferring it through frequency estimation.

4.6.2 Partial Information to Full Information

As briefly discussed earlier, any regret-minimization problem with partial-information feedback can be reduced to the full information case, assuming that we can estimate the loss of all actions by sampling a subset of the actions. The next lemma gives an upper bound on regret in terms of the number of actions sampled, the quality of the estimator (in terms of its range), and the total number of actions.

Lemma 4.6.2. *Let $\mathcal{M}$ be the set of all actions. For any time block (set of consecutive time steps) T' and action $j \in \mathcal{M}$, let $c_{T'}(j)$ be the* average loss *of action j over T'. Assume that $\mathcal{S} \subseteq \mathcal{M}$ is such that by sampling all actions in $\mathcal{S}$, we can compute $\hat{c}_{T'}(j)$ for all $j \in \mathcal{M}$ with the following properties:*

$$\mathbb{E}[\hat{c}_{T'}(j)] = c_{T'}(j) \quad \textit{and} \quad \hat{c}_{T'}(j) \in [-\kappa, \kappa].$$

Then there is an algorithm with loss

$$L_{alg} \leq L_{\min} + O\left(T^{2/3}|\mathcal{S}|^{1/3}\kappa^{1/3}\log^{1/3}(|\mathcal{M}|)\right),$$

where $L_{\min}$ is the loss of the best action in hindsight.

Proof. Our proposed algorithm is as follows. Let $Z = (T^2|\mathcal{S}|^{-2}\kappa \log(|\mathcal{M}|))^{1/3}$. Divide T into Z (roughly) equal blocks, $B_1, \ldots, B_Z$. In each block, pick a uniformly random permutation of $\mathcal{S}$ together with $|\mathcal{S}|$ uniformly random time steps in that block, and assign them to the *exploration phase*. At any time step that is dedicated to exploration, sample the actions in $\mathcal{S}$ in the order they appear in the random permutation. At the end of each block, by the assumptions of the lemma, we receive $\hat{c}_\tau(j)$, which is the average loss of action j during B_τ. We pass this loss information to any regret-minimization algorithm that works in the full information feedback model. At every time step that is not designated for exploration, we use the action that is computed by the full-information algorithm based on the loss from the previous time block.

Next, we compute the regret of the proposed algorithm: Let $q_\tau(\cdot)$ be the fixed probability distribution over actions suggested by the full information algorithm for block B_τ. On the other hand, let $q_t(\cdot)$ be the probability distribution over actions used by the partial information algorithm. That is, during exploitation time steps t in block B_τ, $q_t(\cdot) = q_\tau(\cdot)$, and during the exploration phase $q_t(\cdot)$ refers the action that is being sampled at round t. For any action j, let $\ell_t(j)$ represent the actual loss at round t, and $L_\tau(j)$ and $c_\tau(j)$ indicate the aggregate and average loss of j in block B_τ, respectively. That is

$$L_\tau(j) = \sum_{t \in B_\tau} \ell_t(j) \quad \text{and} \quad c_\tau(j) = \frac{L_\tau(j)}{|B_\tau|}.$$

We have

$$\begin{aligned}
L_{alg} &= \sum_{\tau=1}^{Z} \sum_{t \in B_\tau} \sum_{j \in \mathcal{M}} q_t(j)\ell_t(j) \\
&\leq \sum_{\tau=1}^{Z} \sum_{j \in \mathcal{M}} q_\tau(j) L_\tau(j) + Z \cdot |\mathcal{S}| \quad \text{(Each } j \in \mathcal{S} \text{ is sampled once and has loss} \leq 1) \\
&\leq \sum_{\tau=1}^{Z} \sum_{j \in \mathcal{M}} q_\tau(j)\, \mathbb{E}[\hat{c}_\tau(j)] \left(\frac{T}{Z}\right) + Z \cdot |\mathcal{S}| \quad \text{(Definition of } c_\tau(\cdot) \text{ and unbiasedness of } \hat{c}_\tau(\cdot)) \\
&\leq \frac{T}{Z}\ \mathbb{E}\left[\sum_{\tau=1}^{Z} \sum_{j \in \mathcal{M}} q_\tau(j)\hat{c}_\tau(j)\right] + Z \cdot |\mathcal{S}| \\
&\leq \frac{T}{Z}\ \mathbb{E}\left[\min_j \sum_{\tau=1}^{Z} \hat{c}_\tau(j) + R_{Z,\kappa}\right] + Z \cdot |\mathcal{S}| \quad \text{(Regret bound under full information)} \\
&\leq \frac{T}{Z}\ \left(\min_j \mathbb{E}\left[\sum_{\tau=1}^{Z} \hat{c}_\tau(j)\right] + R_{Z,\kappa}\right) + Z \cdot |\mathcal{S}| \quad \text{(Jensen's Inequality)} \\
&\leq \frac{T}{Z}\ \left(\min_j \sum_{\tau=1}^{Z} c_\tau(j) + R_{Z,\kappa}\right) + Z \cdot |\mathcal{S}| \quad \text{(Unbiasedness of } \hat{c}_\tau(j)) \\
&\leq\ \min_j \sum_{t=1}^{T} \ell_t(j) + \frac{T}{Z} R_{Z,\kappa} + Z \cdot |\mathcal{S}| \quad \text{(Because } \sum_{t=1}^{T} \ell_t(j) = \sum_{\tau=1}^{Z} c_\tau(j)|B_\tau|)
\end{aligned}$$

$$\leq L^T_{\min} + \frac{T}{Z}\sqrt{Z\kappa \log(|\mathcal{M}|)} + Z \cdot |\mathcal{S}|$$
$$\leq L^T_{\min} + O\left(T^{2/3}|\mathcal{S}|^{1/3}\kappa^{1/3}\log^{1/3}(|\mathcal{M}|)\right) \quad \text{(Because } Z = \left(T^2|\mathcal{S}|^{-2}\kappa\log(|\mathcal{M}|)\right)^{1/3}\text{)}$$

□

4.6.3 Creating Unbiased Estimators

Constructing an unbiased loss estimator for each mixed strategy is a pre-requisite for our reduction to the full information case (as in Lemma 4.6.2). Here, we show how such estimators can be constructed for all mixed strategies, by sampling only a small number of mixed strategies.

For any τ, let $f_\tau : \mathbb{R}^k \to \mathbb{R}$ be a function that for any $\mathbf{w} = (w_1, \ldots, w_k)$ returns the number of times attacker types $\theta_1, \ldots, \theta_k$ were active in block B_τ, weighted by coefficients of $\mathbf{w}$. That is

$$f_\tau(\mathbf{w}) = \sum_{j=1}^{k} w_j \sum_{t \in B_\tau} \mathbb{I}(a_t = \theta_j),$$

where $\mathbb{I}$ is an indicator function. For the majority of this section, $\mathbf{w}$ denotes an indicator vector—a binary vector in $\mathbb{R}^k$ indicating the set of attacker types that best-respond to a mixed strategy by attacking a certain target. In this case, $f_\tau(\mathbf{w})$ is the number of times that we encounter these specified attacker types in block B_τ. That is, for some $\Theta' \subseteq \Theta$ and its corresponding indicator vector $\mathbf{w}$, $f_\tau(\mathbf{w})$ is the total number of times attackers in Θ' are active in B_τ. Furthermore, note that $\sum_{t \in B_\tau} \mathbb{I}(a_t = \theta_j)$ is a constant for any fixed τ and j, therefore $f_\tau(\cdot)$ is a linear function.

For any mixed strategy $\mathbf{p} \in P_\sigma$ and any τ, let $c_\tau(\mathbf{p})$ represent the *average* utility of $\mathbf{p}$ against attackers in block B_τ. Then,

$$c_\tau(\mathbf{p}) = \frac{1}{|B_\tau|}\sum_{i=1}^{\mathcal{N}} U_d(i, \mathbf{p}) f_\tau(\mathbb{I}_{(\sigma=i)}),$$

where $\mathbb{I}_{(\sigma=i)}$ is an indicator vector representing all the attackers that respond to a mixed strategy in region $\mathcal{P}_\sigma$ (including $\mathbf{p}$) by attacking i.

Our goal is to construct an unbiased estimator for $c_\tau(\mathbf{p})$ for any $\mathbf{p}$. To do so, we first construct an estimator for $f_\tau(\cdot)$. Since $f_\tau(\cdot)$ is linear in $\mathbb{R}^k$, it suffices to construct an estimator for a spanning set of $\mathbb{R}^k$. To this end, we define $\mathcal{W} = \{\mathbb{I}_{(\sigma=i)} | \text{ for all } i \in \mathcal{N} \text{ and } \sigma \in \Sigma\}$, which is a set of vectors $\mathbb{I}_{(\sigma=i)}$ for any i and σ, each corresponding to attacker types that attack target i in region $\mathcal{P}_\sigma$. Next, we introduce a result by Awerbuch and Kleinberg [27] that helps us in choosing an appropriate basis for $\mathcal{W}$.

Proposition 4.6.3. *([27, Proposition 2.2]) If $\mathcal{W}$ is a compact subset of d-dimensional vector space $\mathcal{V}$, then there exists a set $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\} \subseteq \mathcal{W}$ such that for all $\mathbf{w} \in \mathcal{W}$, $\mathbf{w}$ may be expressed as a linear combination of elements of $\mathcal{B}$ using coefficients in $[-1, 1]$. That is, for all $\mathbf{w} \in \mathcal{W}$, there exist coefficients $\lambda_1, \ldots, \lambda_d \in [-1, 1]$, such that $\mathbf{w} = \sum \lambda_j \mathbf{b}_j$. Such $\mathcal{B}$ is called a* barycentric spanner *for $\mathcal{W}$.*

Consider the set $\mathcal{W}$. First, note that $\mathcal{W}$ is finite, so it is compact and it has a barycentric spanner of size k. Using the methods of Awerbuch and Kleinberg [27], we can construct a barycentric spanner for $\mathcal{W}$ by performing linear optimization over $\mathcal{W}$. Alternatively, for a choice of $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \in \mathcal{W}$, and for all $\mathbf{w} \in \mathcal{W}$, we can solve the following feasibility LP:

$$\forall j \in [k], \ \sum_i \lambda_i b_{i,j} = w_j,$$
$$\forall i, \ -1 \leq \lambda_i \leq +1.$$

Let $\mathcal{B}$ be the barycentric spanner for $\mathcal{W}$ as defined above. For any $\mathbf{b} \in \mathcal{B}$, there must be a mixed strategy $\mathbf{p} \in \mathcal{P}_\sigma$ and target $i \in \mathcal{N}$ such that $\mathbb{I}_{(\sigma=i)} = \mathbf{b}$ (otherwise $\mathbf{b} \notin \mathcal{W}$); call such strategy and target $\mathbf{p}_\mathbf{b}$ and $i_\mathbf{b}$, respectively. We use $\mathbf{p}_\mathbf{b}$ and $i_\mathbf{b}$ for the purpose of creating a loss estimator for $f_\tau(\mathbf{b})$ as follows: In the exploration phase, once at random (based on a chosen random permutation), we play $\mathbf{p}_\mathbf{b}$ and observe target $i_\mathbf{b}$. If $i_\mathbf{b}$ is attacked in response we set $\hat{p}_\tau(\mathbf{b}) = 1$, otherwise $\hat{p}_\tau(\mathbf{b}) = 0$. The next lemma shows that $\hat{p}_\tau(\mathbf{b})|B_\tau|$ is an unbiased estimator for $f_\tau(\mathbf{b})$.

Lemma 4.6.4. *For any* $\mathbf{b} \in \mathcal{B}$, $\mathbb{E}\left[\hat{p}_\tau(\mathbf{b}) \cdot |B_\tau|\right] = f_\tau(\mathbf{b})$.

Proof. Note that $\hat{p}(\mathbf{b}) = 1$ if and only if at the time step that $\mathbf{p}_\mathbf{b}$ was played for the purpose of recording $\mathbf{b}$, target $i_\mathbf{b}$ was attacked. Because $\mathbf{p}_\mathbf{b}$ is played once for the purpose of recording $\mathbf{b}$ uniformly at random over the time steps and the adversarial sequence is chosen before the game play, the attacker that responded to $\mathbf{p}_\mathbf{b}$ is also picked uniformly at random over the time steps. Therefore, $\mathbb{E}[\hat{p}(\mathbf{b})]$ is the probability that a randomly chosen attacker from B_τ responds in a way that is consistent with any one of the attackers who responds by attacking $i_\mathbf{b}$. Formally,

$$\mathbb{E}[\hat{p}_\tau(\mathbf{b})] = \frac{\sum_{i:b_i=1} f_\tau(\mathbf{e}_i)}{|B_\tau|} = \frac{f_\tau(\mathbf{b})}{|B_\tau|}.$$

□

Since $\mathcal{W} \subseteq \{0,1\}^k$, the rank of $\mathcal{W}$ is at most k. Let $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ be the barycentric spanner for $\mathcal{W}$. For any $\mathbf{w} \in \mathcal{W}$, let $\boldsymbol{\lambda}(\mathbf{w})$ be the representation of $\mathbf{w}$ in basis $\mathcal{B}$. That is, $\sum_{j=1}^k \lambda_j(\mathbf{w})\, \mathbf{b}_j = \mathbf{w}$. Now, consider any mixed strategy $\mathbf{p}$ and let σ be such that $\mathbf{p} \in \mathcal{P}_\sigma$. Let

$$\hat{c}_\tau(\mathbf{p}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j(\mathbb{I}_{\sigma=i})\, \hat{p}_\tau(\mathbf{b}_j) U_d(i, \mathbf{p}). \tag{4.2}$$

The next lemma shows that for all $\mathbf{p}$, $\hat{c}_\tau(\mathbf{p})$ is indeed an unbiased estimator of $c_\tau(\mathbf{p})$ with a small range.

Lemma 4.6.5. *For any mixed strategy* $\mathbf{p}$, $\mathbb{E}[\hat{c}_\tau(\mathbf{p})] = c_\tau(\mathbf{p})$ *and* $\hat{c}_\tau(\mathbf{p}) \in [-nk, nk]$.

Proof. Let σ be such that $\mathbf{p} \in \mathcal{P}_\sigma$. We have,

$$\mathbb{E}[\hat{c}_\tau(\mathbf{p})] = \mathbb{E}\left[\sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j(\mathbb{I}_{\sigma=i})\, \hat{p}_\tau(\mathbf{b}_j) U_d(i, \mathbf{p})\right]$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j(\mathbb{I}_{\sigma=i})\ \mathbb{E}[\hat{p}_\tau(\mathbf{b}_j)]\ U_d(i,\mathbf{p}) && \text{(linearity of expectation)} \\
&= \sum_{i=1}^{n} U_d(i,\mathbf{p}) \sum_{j=1}^{k} \frac{\lambda_j(\mathbb{I}_{\sigma=i})\ f_\tau(\mathbf{b}_j)}{|B_\tau|} && \text{(by Lemma 4.6.4)} \\
&= \sum_{i=1}^{n} \frac{U_d(i,\mathbf{p})}{|B_\tau|} f_\tau\left(\sum_{j=1}^{k} \lambda_j(\mathbb{I}_{\sigma=i})\ \mathbf{b}_j\right) && \text{(by linearity of } f_\tau) \\
&= \sum_{i=1}^{n} \frac{U_d(i,\mathbf{p})}{|B_\tau|} f_\tau(\mathbb{I}_{(\sigma=i)}) && \text{(by definition of } \boldsymbol{\lambda}(\cdot)) \\
&= c_\tau(\mathbf{p}).
\end{aligned}
$$

Since $\mathcal{B}$ is barycentric, for any $\mathbf{w} \in \mathcal{W}$, $\lambda_j(\mathbf{w}) \in [-1,1]$. Moreover, $\hat{p}_\tau(\cdot) \in \{0,1\}$, and $U_d(i,\mathbf{p}) \in [-1,1]$. So,

$$
\hat{c}_\tau(\mathbf{p}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j(\mathbb{I}_{\sigma=i})\ \hat{p}_\tau(\mathbf{b}_j) U_d(i,\mathbf{p}) \in [-nk, nk].
$$

□

4.6.4 Putting It All Together

Using the machinery developed in previous subsections, we can now proceed to prove the theorem.

Proof of Theorem 4.6.1. We use Algorithm 4.1 along with Proposition 4.3.1 as a black-box full-information regret minimization algorithm. We use $\mathcal{E}$ as the set of mixed strategies described in Section 4.4.

Our algorithm divides the timeline to $Z = n\,(T^2 \log nk)^{1/3}$ equal intervals, $B_1, \ldots, B_Z$. The initial distribution over the set of mixed strategies $\mathcal{E}$ is the uniform distribution.

In each block, we pick a random permutation π over $\mathcal{B}$ together with k time steps in that block and mark them for exploration. At the j^{th} time step that is dedicated to exploration, we play mixed strategy $\mathbf{p}_{b_{\pi(j)}}$ and observe target $i_{b_{\pi(j)}}$. If $i_{b_{\pi(j)}}$ is attacked, then we assign $\hat{p}_\tau(b_{\pi(j)}) = 1$ otherwise $\hat{p}_\tau(b_{\pi(j)}) = 0$. At any time step that is not set for exploration, we choose a mixed strategy at random from the current distribution over $\mathcal{E}$.

At the end of each block, we compute $\hat{c}_\tau(\mathbf{p})$ for all $\mathbf{p} \in \mathcal{E}$, using Equation 4.2. We then pass this loss information to any regret-minimization algorithm that works in the full information feedback model, and update the default distribution based on its outcome.

Using Lemma 4.6.5, $\hat{c}_\tau(\mathbf{p})$ is an unbiased estimator of the average loss of action $\mathbf{p}$ during B_τ. This unbiased estimator is passed to the full-information regret minimization algorithm. By Lemma 4.6.2, we have

$$
\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - \mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] \leq O\left(T^{2/3} |\mathcal{B}|^{1/3} (nk)^{1/3} \log^{1/3}(|\mathcal{E}|)\right)
$$

Algorithm 4.1: REPEATED SECURITY GAMES WITH PARTIAL INFORMATION

Input: Black-box access to an algorithm that satisfies Proposition 4.3.1, FULL-INFORMATION$(\cdot)$, which takes as input the loss of all actions and produces a distribution $\mathbf{q}$ over them.
$Z \leftarrow n\left(T^2 \log nk\right)^{1/3}$.
Create set $\mathcal{W} = \{\mathbb{I}_{(\sigma=i)}|$ for all $i \in \mathcal{N}$ and $\sigma \in \Sigma\}$.
Find a barycenteric spanner $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ for $\mathcal{W}$. For every $\mathbf{b} \in \mathcal{B}$ such that $\mathbf{b} = \mathbb{I}_{(\sigma=i)}$ for some i and σ, let $i_{\mathbf{b}} \leftarrow i$ and $\mathbf{p}_{\mathbf{b}}$ be any mixed strategy in $\mathcal{P}_\sigma$.
For all $\mathbf{w} \in \mathcal{W}$ let $\boldsymbol{\lambda}(\mathbf{w})$ be the representation of $\mathbf{w}$ in basis $\mathcal{B}$. That is $\sum_{j=1}^{k} \lambda_j(\mathbf{w})\, \mathbf{b}_j = \mathbf{w}$.

Let $\mathbf{q}_1$ be the uniform distribution over $\mathcal{E}$.
for $\tau = 1, \ldots, Z$ **do**
  Choose a random permutation π over $[k]$ and $t_1, \ldots, t_k$ time steps at random from $[T/Z]$.
  for $t = (\tau - 1)(T/Z) + 1, \ldots, \tau(T/Z)$ **do**
    if $t = t_j$ for some $j \in [k]$ **then**
      Play $\mathbf{p}_t \leftarrow \mathbf{p}_{\mathbf{b}_{\pi(j)}}$
      If $i_{\mathbf{b}_{\pi(j)}}$ is attacked, then $\hat{p}_\tau(\mathbf{b}_{\pi(j)}) \leftarrow 1$, otherwise $\hat{p}_\tau(\mathbf{b}_{\pi(j)}) \leftarrow 0$.
    else
      Play $\mathbf{p}_t$ at random from distribution $\mathbf{q}_\tau$.
    end if
  end for
  for all $\mathbf{p} \in \mathcal{E}$ and all σ such that $\mathbf{p} \in \mathcal{P}_\sigma$ **do**

$$\hat{c}_\tau(\mathbf{p}) \leftarrow \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j(\mathbb{I}_{\sigma=i})\, \hat{p}_\tau(\mathbf{b}_j) U_d(i, \mathbf{p}).$$

  end for
  Call FULL-INFORMATION$(\hat{c}_\tau)$. And receive $\mathbf{q}_{\tau+1}$ as a distribution over all mixed strategies in $\mathcal{E}$.
end for

$$\begin{aligned} &\in O\left(T^{2/3} k^{1/3} (nk)^{1/3} (n^2 k \log(nk))^{1/3}\right) \\ &\in O\left(T^{2/3}\, nk\, \log^{1/3}(nk)\right). \end{aligned}$$

□

4.7 Lower Bound

Can we be truly blind to the types of attackers that the defender might encounter? As mentioned before, the set of all attacker types, Θ, is known to the defender. But what if we allow Θ to include all possible types of attackers? In this section, we show that with no prior knowledge regarding the possible types of attackers that the defender might encounter, it is impossible to

design a no-regret algorithm. This is formalized in our final theorem that shows that for any T there is a game with at most 2^{T+1} attacker types such that any online algorithm experiences regret that is linear in T.

Theorem 4.7.1. *For any T there is a repeated security game in the full-information feedback setting with a set Θ of attacker types such that $|\Theta| < 2^{T+1}$ and any online algorithm that at time t (possibly at random) returns strategy $\mathbf{p}_t$ has expected utility*

$$\mathbb{E}\left[\sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}_t), \mathbf{p}_t)\right] \leq \sum_{t=1}^{T} U_d(b_{a_t}(\mathbf{p}^*), \mathbf{p}^*) - \frac{T}{2},$$

where the expectation is over the algorithm's internal randomization.

Proof. Consider a security game in which $\mathcal{N} = \{1, 2, 3\}$, and the defender has one resource that can defend any one target at a time. Let the defender's utility be $U_d(i, \mathbf{p}) = -1$ for $i \in \{1, 3\}$ and $U_d(2, \mathbf{p}) = 0$. Let all attackers break ties in lexicographic order.

Because the defender prefers target 2 to be attacked, for any coverage probability $\mathbf{p}$, reducing p_2 to 0 leads to a valid coverage probability that will not decrease the defender's payoff. So, without loss of generality we restrict the defender's actions to the coverage probabilities $\mathbf{p} = [p, 0, 1-p]$.

Next, we define a set Θ of attackers. To do so, we first show that certain best-response functions are valid, i.e. there is an attacker—defined by its utility function—that responds according to that best-response function. The next lemma allows us to define the attackers by their best-response functions, thereby significantly simplifying our analysis.

Lemma 4.7.2. *For any $0 \leq r_1 \leq r_2 \leq 1$ and any $\mathbf{p} = [p, 0, 1-p]$, there exists an attacker type θ_j (defined by its utility function) such that*

$$b_{\theta_j}(\mathbf{p}) = \begin{cases} 1 & \text{if } p \in [0, r_1] \\ 2 & \text{if } p \in (r_1, r_2] \\ 3 & \text{if } p \in (r_2, 1] \end{cases}$$

Proof. Let $\alpha = \max\{\frac{1-r_1}{r_1}, \frac{r_2}{1-r_2}\}$. Let θ_j be an attacker defined as follows.

$$u^c_{\theta_j}(i) = \begin{cases} \frac{-(1-r_1)}{r_1\alpha} & \text{if } i = 1 \\ 0 & \text{if } i = 2 \\ \frac{-r_2}{(1-r_2)\alpha} & \text{if } i = 3 \end{cases} \quad \text{and} \quad u^u_{\theta_j}(i) = \begin{cases} 0 & \text{if } i = 2 \\ \frac{1}{\alpha} & \text{otherwise} \end{cases}$$

We show that attacker θ_j's best-response has the desired properties. For $p \in [0, r_1]$,

$$\begin{aligned} U_{\theta_j}(1, \mathbf{p}) &= \frac{1}{\alpha}\left(p\frac{-(1-r_1)}{r_1} + (1-p)\right) \geq \frac{1}{\alpha}(-(1-r_1) + (1-p)) \geq 0, \\ U_{\theta_j}(2, \mathbf{p}) &= 0, \\ U_{\theta_j}(3, \mathbf{p}) &= \frac{1}{\alpha}\left((1-p)\frac{-r_2}{1-r_2} + p\right) \leq \frac{1}{\alpha}(-r_2 + p) \leq 0, \end{aligned}$$

so, $b_{\theta_j}(\mathbf{p}) = 1$ for $p \in [0, r_1]$. For $p \in (r_1, r_2]$,

$$U_{\theta_j}(2, \mathbf{p}) = 0 > \frac{1}{\alpha}\left(-(1-r_1) + (1-p)\right) > \frac{1}{\alpha}\left(p\frac{-(1-r_1)}{r_1} + (1-p)\right) = U_{\theta_j}(1, \mathbf{p}),$$
$$U_{\theta_j}(2, \mathbf{p}) = 0 \geq \frac{1}{\alpha}\left(-r_2 + p\right) \geq \frac{1}{\alpha}\left((1-p)\frac{-r_2}{1-r_2} + p\right) = U_{\theta_j}(3, \mathbf{p}),$$

so, $b_{\theta_j}(\mathbf{p}) = 2$ for $p \in (r_1, r_2]$. For $p \in (r_2, 1]$,

$$U_{\theta_j}(1, \mathbf{p}) = \frac{1}{\alpha}\left(p\frac{-(1-r_1)}{r_1} + (1-p)\right) < \frac{1}{\alpha}\left(-(1-r_1) + (1-p)\right) < 0,$$
$$U_{\theta_j}(2, \mathbf{p}) = 0,$$
$$U_{\theta_j}(3, \mathbf{p}) = \frac{1}{\alpha}\left((1-p)\frac{-r_2}{1-r_2} + p\right) > \frac{1}{\alpha}\left(-r_2 + p\right) > 0,$$

so, $b_{\theta_j}(\mathbf{p}) = 3$. Therefore, the attacker defined by the above utility function has the desired best-response. □

Next, we recursively define $2^{T+1} - 2$ attacker types. We use $r_1 \leq r_2$ as defined in Lemma 4.7.2 to represent the best response of an attacker. We represent attacker types by binary strings. Let

$$\text{Attacker } \theta_0 : r_1^0 = \frac{1}{2}, r_2^0 = 1$$
$$\text{Attacker } \theta_1 : r_1^1 = 0, r_2^1 = \frac{1}{2}.$$

For any $x \in \{0,1\}^{<T}$ define

$$\text{Attacker } \theta_{0x} : r_1^{0x} = \frac{r_1^x + r_2^x}{2}, r_2^{0x} = r_2^x$$
$$\text{Attacker } \theta_{1x} : r_1^{1x} = r_1^x, r_2^{1x} = \frac{r_1^x + r_2^x}{2}$$

This representation allows us to think of the choices of the adversary as a chain of post-fixes of a T-bit binary number. That is, the attacker chooses a T-bit binary string and plays its post-fixes in order.

Next, we formulate our problem in terms of decision trees. Consider a complete binary tree. The adversary's sequence of attackers indicates a root-to-leaf path in this tree that corresponds to his choice of the T-bit string. The defender's online decision at every step of the game is to choose a distribution over $p \in (r_1^{0x}, r_2^{0x}]$ or $p \in (r_1^{1x}, r_2^{1x}]$, having already observed string x. Since for all x, $(r_1^{0x}, r_2^{0x}] \cap (r_1^{1x}, r_2^{1x}] = \emptyset$, these two choices represent two disjoint events. Therefore, we can represent the defender's online decision at node x as a distribution on nodes $0x$ or $1x$. Using Lemma 4.7.2, if both the defender and attacker land on the same node, the defender receives a penalty of 0 and otherwise receives a penalty of 1 (utility -1). To summarize, the online algorithm corresponds to a distribution over all possible decision trees; the adversary chooses a root-to-leaf path; and expected utility is calculated as above.

By Yao's Minmax Principle, an upper bound on the expected utility of the optimal randomized decision tree against the worst deterministic sequence (root-to-leaf path) can be obtained by constructing a specific distribution over sequences, and reasoning about the expected utility of the best *deterministic* decision tree against this distribution. Let this distribution be the uniform distribution over all T-bit strings. That is, at step x the adversary chooses attackers $0x$ and $1x$ each with probability $\frac{1}{2}$. Then for any fixed decision tree, at every node the algorithm only has $\frac{1}{2}$ probability of matching the adversary's choice. So, the defender receives an expected penalty of $\frac{1}{2}$. We conclude that for any randomized online algorithm there exists a sequence of attackers—corresponding to a T-bit string—such that the algorithm's expected utility is at most $-\frac{T}{2}$.

To complete the proof, we claim that for any sequence of attackers corresponding to a T-bit string, there is a mixed strategy that would cause each attacker in the sequence to attack target 2. This is true because for every x and y, $(r_1^{yx}, r_2^{yx}] \subset (r_1^x, r_2^x]$, and therefore if θ_x is the attacker at step T, choosing $p^* \in (r_1^x, r_2^x]$ would also place p in the interval corresponding to any previous attacker. By Lemma 4.7.2, the best response of each attacker in the sequence to the strategy $\mathbf{p}^* = [p^*, 0, 1-p^*]$ is to attack target 2. It follows that $\mathbf{p}^*$ has overall utility 0 to the defender, and hence the regret is at least $\frac{T}{2}$. □

4.8 Discussion

Extension to general Stackelberg games The approach presented in this chapter easily extends to general repeated Stackelberg games, which can be represented as follows. There are k matrices of size $n \times m$, with the same payoffs for the row player, but possibly different payoffs for the column player. This set of matrices is known to the players. At each time step, a game matrix is chosen but remains unknown to the row player. The row player then chooses a probability distribution over the rows of the matrix and the column player, having observed this distribution, chooses the column with the best expected payoff. An online algorithm should guarantee to the row player good payoff against any adversarially selected sequence of such matrices.

Note that a repeated SSG is captured by the foregoing framework: in each matrix, each row represent a pure strategy of the defender (there may be exponentially many rows) and each column represents a target. Then, for row i and column j, if target j is covered in the i^{th} strategy then the row and column payoff are $u_d^c(i)$ and $u_a^c(i)$, and otherwise $u_d^u(i)$ and $u_a^u(i)$, respectively.

To see how our methodology extends, note that each of the k matrices can be used to decompose the space of all probability distributions (over the rows) into m convex regions, where in each region the best response is a fixed column. For the full information feedback model, the set of all extreme points in the intersections of these regions leads to an algorithm whose regret is polynomial in m, n and k, and sublinear in T. Similarly, for the partial information feedback model, our approach for inferring the frequency of each matrix by only observing the best response carries over to the general Stackelberg games. However, we focused on SSGs in order to obtain regret bounds that are polynomial in the number of targets and attacker types—the foregoing bounds for general Stackelberg games would translate to bounds that are exponential in the number of targets.

On the power of adversary In our work, we assume that the sequence of attackers is chosen adversarially before the game starts. That is, the adversary's choice of attacker is *oblivious* to the history. It is worth noting that our upper bound on regret in the full-information feedback setting holds for a much more powerful adversary; an *adaptive* adversary who chooses an attacker at time t by first observing the defender's mixed strategies $\mathbf{p}_1, \ldots, \mathbf{p}_{t-1}$, and the defender's distribution (determined by the internal randomness of the online algorithm) over mixed strategies at steps $1, \ldots, t$. It would be interesting to know whether there are no-regret algorithms, with polynomial dependence on the number of targets and types, for adaptive adversaries and partial-information feedback.

On the benefits of laziness Our regret analysis holds when used in conjunction with any regret-minimization algorithm that satisfies Proposition 4.3.1. Nevertheless, some algorithms provide additional properties that may prove useful in practice. Specifically, when used with *Follow the Lazy Leader*, our algorithm for the full information setting uses one mixed strategy for a long period of time before switching to another (expected length $\tilde{O}(\sqrt{T})$). Informally speaking, this allows attackers enough time to conduct surveillance, observe the mixed strategy, and then best-respond to it. Therefore, even if the attacker's response to a new mixed strategy is unreliable or irrational at first (not representative of his best response), the defender can still guarantee a no-regret outcome.

The extremely partial information model Previous work has mostly assumed that the defender is able to monitor all targets simultaneously and detect an attack on any one of them at all times [178, 203]. In contrast, our algorithm for the partial information setting only requires the ability to detect whether or not an attack occurs on at most *one chosen target* at a time. This feature may prove useful in domains where simultaneous observation of targets is impractical. For example, in wildlife protection applications, patrols and unmanned aerial vehicles (UAVs) can detect signs of poaching only in very specific areas.

4.9 Subsequent Works

Following the initial publication of the results that appear in this chapter, Daskalakis and Syrgkanis [94] considered online learning in the general case and with applications to auctions design and online Stackelberg games. They showed that in any full information online learning problem where the adversary has d actions, there is an algorithm with regret $O(\sqrt{dT})$. In the context of online Stackelberg games with full information, Daskalakis and Syrgkanis [94] obtained an improved regret of $O(\sqrt{kT})$ regardless of the number of targets, n. Subsequently, in our recent work [113], we follow up on the work of Daskalakis and Syrgkanis [94] and show that one can improve the regret of general online learning problems further down to $\sqrt{d'T}$ for some $d' \ll d$, when the set of d actions of the adversary and the set of actions of the learner demonstrate a certain structural property parameterized by $d' \ll d$. We present these results in the next chapter.

Chapter 5

Oracle-Efficient Online Learning and Auction Design

5.1 Introduction

In this chapter, we consider the computational aspect of online learning. Online learning algorithms have been designed for a variety of problems, such as multi-attacker Stackelberg games (Chapter 4), online marketplaces [31, 52, 73, 239], and communication networks [27]. The environments in these applications are constantly evolving, requiring continued adaptation of these systems. General-purpose online learning algorithms robustly address this challenge, with performance guarantees that hold even when the environment is adversarial. However, these general purpose algorithms (often with information-theoretically optimal guarantees) are computationally inefficient when the action space is exponential in the natural problem representation [124]. In this chapter, we introduce online algorithms that can be implemented efficiently and with little effort for those exponentially large actions spaces that demonstrate some structural properties.

This goal is not achievable without some assumptions on the problem structure. Since an online optimization problem is at least as hard as the corresponding offline optimization problem [71, 94], a minimal assumption is the existence of an algorithm that returns a near-optimal solution to the offline problem. We assume, without loss of generality, that our learner has access to such an offline algorithm, which we call *an offline optimization oracle*. This oracle, for any (weighted) history of choices by the environment, returns an action of the learner that (approximately) maximizes the learner's reward. We seek to design *oracle-efficient learners*, that is, learners that run in polynomial time, with each oracle call counting $O(1)$.

An oracle-efficient learning algorithm can be viewed as a reduction from the online to the offline problem, providing conditions under which the online problem is not only as hard, but also as easy as the offline problem, and thereby offering computational equivalence between online and offline optimization. Apart from theoretical significance, reductions from online to offline optimization are also practically important. For example, if one has already developed and implemented a Bayesian optimization procedure which optimizes against a static stochastic environment, then our algorithm offers a black-box transformation of that procedure

into an adaptive optimization algorithm with provable learning guarantees in non-stationary, non-stochastic environments. Even if the existing optimization system does not run in worst-case polynomial time, but is rather a well-performing fast heuristic, a reduction to offline optimization will leverage any expert domain knowledge that went into designing the heuristic, as well as any further improvements of the heuristic or even discovery of polynomial-time solutions.

Recent work of Hazan and Koren [156] shows that oracle-efficient learning in adversarial environments is not achievable in general, while leaving as open the problem of identifying the properties under which oracle-efficient online learning may be possible. Specifically, we introduce a general purpose algorithm called *Generalized Follow-the-Perturbed-Leader* (Generalized FTPL) and derive sufficient conditions under which this algorithm yields oracle-efficient online learning. Our results are enabled by providing a new way of adding *regularization* so as to *stabilize* optimization algorithms in general optimization settings. The latter could be of independent interest beyond online learning. Our approach unifies and extends previous approaches to oracle-efficient learning, including the Follow-the-Perturbed Leader (FTPL) approach introduced by Kalai and Vempala [169] for linear objective functions, and its generalizations to submodular objective functions [155], adversarial contextual learning [254], and learning in simultaneous second-price auctions [94]. Furthermore, our sufficient conditions draw a strong connection between the notion of a universal identification set of Goldman et al. [131] and oracle-efficient learnability.

The second main contribution of our work is to introduce a new framework for the problem of adaptive auction design for revenue maximization and to demonstrate the power of Generalized FTPL through several applications in this framework. Traditional auction theory assumes that the valuations of the bidders are drawn from a population distribution which is known, thereby leading to a Bayesian optimization problem. The knowledge of the distribution by the seller is a strong assumption. Recent work in algorithmic mechanism design [83, 103, 210, 237] relaxes this assumption by solely assuming access to a set of samples from the distribution. In this work, we drop any distributional assumptions and introduce the adversarial learning framework of *online auction design*. On each round, a learner adaptively designs an auction rule for the allocation of a set of resources to a fresh set of bidders from a population.[1] The goal of the learner is to achieve average revenue at least as large as the revenue of the best auction from some target class. Unlike the standard approach to auction design, initiated by the seminal work of Myerson [212], our approach is devoid of any assumptions about a prior distribution on the valuations of the bidders for the resources at sale. Instead, similar to an agnostic approach in learning theory, we incorporate prior knowledge in the form of a target class of auction schemes that we want to compete with. This is especially appropriate when the auctioneer is restricted to using a particular design of auctions with power to make only a few design choices, such as deciding the reserve prices in a second price auction. A special case of our framework is considered in the recent work of Roughgarden and Wang [239]. They study online learning of the class of single-item second-price auctions with bidder-specific reserves, and give an algorithm with performance that approaches a constant factor of the optimal revenue in hindsight. We go

[1]Equivalently, the set of bidders on each round can be the same as long as they are myopic and optimize their utility separately in each round. Using extension to contextual learning (See full version of our results in [113]), this approach can also be applied when the learner's choice of auction is allowed to depend on features of the arriving set of bidders, such as demographic information.

well beyond this specific setting and show that our Generalized FTPL can be used to optimize over several standard classes of auctions including VCG auctions with bidder-specific reserves and the level auctions of Morgenstern and Roughgarden [210], achieving low additive regret to the best auction in the class.

In the remainder of this section, we describe our main results in more detail and then discuss several extensions and applications of these results, including (1) learning with constant-factor approximate oracles (e.g., using Maximal-in-Range algorithms [216]); (2) regret bounds with respect to stronger benchmarks for the case in which the environment is not completely adversarial but follows a fast-mixing Markov process.

Our work contributes to two major lines of work on the design of efficient and oracle-efficient online learning algorithms [5, 94, 112, 155, 156, 167, 169, 228, 255] and the design of auctions using machine learning tools [52, 73, 83, 103, 180, 210].

5.1.1 Oracle-Efficient Learning with Generalized FTPL

We consider the following online learning problem. On each round $t = 1, \ldots, T$, a learner chooses an action x_t from a finite set $\mathcal{X}$, and an adversary chooses an action y_t from a set $\mathcal{Y}$. The learner then observes y_t and receives a payoff $f(x_t, y_t) \in [0, 1]$, where the function f is fixed and known to the learner. The goal of the learner is to obtain low expected regret with respect to the best action in hindsight, i.e., to minimize

$$\textsc{Regret} := \mathbb{E}\left[\max_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right],$$

where the expectation is over the randomness of the learner.[2] We desire algorithms, called *no-regret algorithms*, for which this regret is sublinear in the time horizon T.

Our algorithm takes its name from the seminal Follow-The-Perturbed-Leader (FTPL) algorithm of Kalai and Vempala [169]. FTPL achieves low regret, $O(\sqrt{T \log(|\mathcal{X}|)})$, by independently perturbing the historical payoff of each of the learner's actions and choosing on each round the action with the highest perturbed payoff. However, this approach is inefficient when the action space is exponential in the natural representation of the learning problem, because it requires creating $|\mathcal{X}|$ independent random variables.[3] Moreover, because of the form of the perturbation, the optimization of the perturbed payoffs cannot be performed by the offline optimization oracle for the same problem, but instead it requires a "perturbed" optimization oracle. We overcome both of these challenges by, first, generalizing FTPL to work with perturbations that can be compactly represented and are thus not necessarily independent across different actions (*sharing randomness*), and, second, by implementing such perturbations via synthetic histories of adversary actions (*implementing randomness*).

[2] To simplify exposition, we assume that the adversary is oblivious, i.e., that the sequence $y_1, \ldots, y_T$ is chosen in advance, though our results generalize to adaptive adversaries using standard techniques [94, 163].

[3] If payoffs are linear in some low-dimensional representation of $\mathcal{X}$ then the number of variables needed is equal to this dimension. But for non-linear payoffs, $|\mathcal{X}|$ variables are required.

Sharing randomness Our Generalized FTPL begins by drawing a random vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ of some small size N, with components α_j drawn independently from a dispersed distribution D. The payoff of each of the learner's actions is perturbed by a linear combination of these independent variables, as prescribed by a *perturbation translation matrix* $\boldsymbol{\Gamma}$ of size $|\mathcal{X}| \times N$, with entries in $[0, 1]$. Let $\boldsymbol{\Gamma}_x$ denote the row of $\boldsymbol{\Gamma}$ corresponding to x. On each round t, the algorithm outputs an action x_t that (approximately) maximizes the perturbed historical performance. In other words, x_t is chosen such that for all $x \in \mathcal{X}$,

$$\sum_{\tau=1}^{t-1} f(x_t, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_t} \geq \sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon$$

for some fixed optimization accuracy $\epsilon \geq 0$. See Algorithm 5.1 in Section 5.2 for a full specification.

We show that Generalized FTPL is no-regret as long as ϵ is sufficiently small and the translation matrix $\boldsymbol{\Gamma}$ satisfies an *admissibility* condition. This condition requires the rows of $\boldsymbol{\Gamma}$ to be (sufficiently) distinct so that each action's perturbation uses a different weighted combination of the low-dimensional noise. To the best of our knowledge, the approach of using an arbitrary matrix to induce shared randomness among actions of the learner is novel. See Theorem 5.2.5 for a formal statement of this result.

Theorem 5.2.5 (informal). *A translation matrix is (κ, δ)-admissible if any two rows of the matrix are distinct, the number of different values within a column is at most κ, and the minimum non-zero difference between any two values within a column is at least δ. Generalized FTPL with a (κ, δ)-admissible matrix $\boldsymbol{\Gamma}$ and an appropriate uniform distribution as D achieves regret $O(N\sqrt{T\kappa/\delta} + \epsilon T)$.*

A technical challenge here is to show that the randomness induced by $\boldsymbol{\Gamma}$ on the set of actions $\mathcal{X}$ *stabilizes* the algorithm, i.e., the probability that $x_t \neq x_{t+1}$ is small. We use the admissibility of $\boldsymbol{\Gamma}$ to guide us through the analysis of stability. In particular, we consider how each column of $\boldsymbol{\Gamma}$ partitions actions of $\mathcal{X}$ to a few subsets (at most κ) based on their corresponding entries in that column. Admissibility implies that the algorithm is stable as a whole, if for each column the partition to which an action belongs remains the same with probability close to 1. This allows us to decompose the stability analysis of the algorithm as a whole to the analysis of stability across partitions of each column. At the column level, stability of the partition between two time-steps follows by showing that a switch between partitions happens only if the perturbation α_j corresponding to that column falls into a small sub-interval of its support. The latter probability is small if the distribution is sufficiently dispersed. This final argument is similar in nature to the reason why perturbations lead to stability in the original FTPL algorithm of [169].

Implementing randomness To ensure oracle-efficient learning, we additionally need the property that the induced action-level perturbations can be simulated by a (short) synthetic history of adversary actions. This allows us to avoid working with $\boldsymbol{\Gamma}$ directly, or even explicitly writing it down. This requirement is captured by our *implementability* condition, which states that each column of the translation matrix essentially corresponds to a scaled version of the

expected reward of the learner on some distribution of adversary actions. See Theorem 5.2.9 for a formal statement of this result.

Theorem 5.2.9 (informal). *A translation matrix is implementable if each column corresponds to a scaled version of the expected reward of the learner against some small-supported distribution of actions of the adversary. Generalized FTPL with an implementable translation matrix can be implemented with one oracle call per round and running time polynomial in N, T, and the size of the support of the distribution implementing the translation matrix. Oracle calls count $O(1)$ in the running time.*

For some learning problems, it is easier to first construct an implementable translation matrix and argue about its admissibility; for others, it is easier to construct an admissible matrix and argue about its implementability. We will see examples of each in the applications below, exhibiting the versatility of our conditions.

The following is one consequence of our theorems that is particularly useful for obtaining oracle-efficient no-regret algorithms (see Theorems 5.2.5 and 5.2.9 for more general statements):

> If there exist N adversary actions such that any pair of learner's actions yields different rewards for at least one of these N actions, then Generalized FTPL has regret $O(N\sqrt{T}/\delta)$ and runs in time $\mathrm{poly}(N, T)$ where δ is the smallest difference between distinct rewards on any one of the N actions.

The aforementioned results establish a reduction from online optimization to offline optimization. When the offline optimization problem can indeed be solved in polynomial time, these results imply that the online optimization problem can also be solved in polynomial time. See Corollary 5.2.10 for the associated runtime.

5.1.2 Main Application: Online Auction Design

In many applications of auction theory, including electronic marketplaces, a seller repeatedly sells an item or a set of items to a population of buyers, with a few arriving for each auction. In such cases, the seller can optimize his auction design in an online manner, using historical data consisting of observed bids. We consider a setting in which the seller would like to use this historical data to select an auction from a fixed target class. For example, a seller in sponsored-search auctions might be limited by practical constraints to consider only second-price auctions with bidder-specific reserves. The seller can optimize the revenue by using the historical data for each bidder to set these reserves. Similarly, a seller on eBay may be restricted to set a single reserve price for each item. Here, the seller can optimize the revenue by using historical data from auctions for similar goods to set the reserves for new items. In both cases, the goal is to leverage the historical data to pick an auction on each round in such a way that the seller's overall revenue compares favorably with the optimal auction from the target class.

More formally, on round $t = 1, \ldots, T$, n bidders arrive with a vector of bids (or equivalently, valuations, since we assume the auctions used are truthful) $\mathbf{v}_t \in \mathcal{V}^n$. We allow these valuations to be arbitrary, e.g., chosen by an adversary. Prior to observing the bids, the auctioneer commits to an auction a_t from a class of truthful auctions $\mathcal{A}$. The goal of the auctioneer is to achieve a revenue that, in hindsight, is very close to the revenue that would have been achieved by the best

fixed auction in class $\mathcal{A}$ if that auction were used on all rounds. In other words, the auctioneer aims to minimize the expected regret

$$\mathbb{E}\left[\max_{a\in\mathcal{A}}\sum_{t=1}^{T}\mathrm{Rev}(a,\mathbf{v}_t)-\sum_{t=1}^{T}\mathrm{Rev}(a_t,\mathbf{v}_t)\right],$$

where $\mathrm{Rev}(a,\mathbf{v})$ is the revenue of auction a on bid profile $\mathbf{v}$ and the expectation is over the actions of the auctioneer.

This problem can easily be cast in our oracle-efficient online learning framework. The learner's action space is the set of target auctions $\mathcal{A}$, while the adversary's action space is the set of bid or valuation vectors $\mathcal{V}^n$. Finally, the offline oracle is a revenue maximization oracle which computes an (approximately) optimal auction within the class $\mathcal{A}$ given a set of valuation vectors. Using the Generalized FTPL with appropriate matrices $\mathbf{\Gamma}$, we provide the first oracle-efficient no-regret algorithms for several commonly studied auction classes:

– Vickrey-Clarkes-Groves (VCG) auctions with bidder-specific reserve prices in single-dimensional matroid auction settings, which are known to achieve half the revenue of the optimal auction in i.i.d. settings under some conditions [152];

– envy-free item pricing mechanisms in combinatorial markets with unlimited supply, often studied in the static Bayesian setting [31, 141];

– single-item level auctions, introduced by Morgenstern and Roughgarden [210], who show that these auctions approximate, to an arbitrary accuracy, the Myerson auction [212], which is known to be optimal for the Bayesian independent-private-value setting.

The crux of our approach is designing admissible and implementable matrices. In the case of VCG with bidder-specific reserves and envy-free item pricing auctions, we show how one can implement an (obviously admissible) matrix $\mathbf{\Gamma}$, where each row corresponds to the concatenation of the binary representations of the reserves of each bidder or the prices of each item, respectively. We show that, surprisingly, any perturbation on the auction revenues that is a linear function of this bit representation can be simulated by a distribution of bidder valuations (see Figure 5.1 for an example such construction). For the case of level auctions, our challenge is to show that an (obviously implementable) matrix $\mathbf{\Gamma}$ whose columns refer to a specific small subset of bid profiles is admissible. The hard part of this construction is identifying a small set of bidder valuation vectors, such that any two different level auctions yield different revenues on at least one of these valuation vectors.

Table 5.1 summarizes the regret of our oracle-efficient algorithms, as well as the computational efficiency assuming oracle calls take $O(1)$ computation. All our results perform a single oracle call per iteration, so T oracle calls in total. Note that these results demonstrate an efficient reduction from the online problem to the offline problem.

While in theory, the auction classes discussed in this table do not have a worst-case polynomial time algorithm for solving the offline problem, in practice there are fast running algorithms, e.g., highly optimized Integer Program solvers, that can perform these computations. Hence, the key practical appeal of online to offline reduction is the fact that it enables one to tap into

Auction Class	Regret	Oracle-Based Complexity	Section
VCG with bidder-specific reserves, s-unit	$O(ns\log(T)\sqrt{T})$	$O(nT^{3/2}\log(T))$	5.3.1
envy free k-item pricing	$O(nk\log(T)\sqrt{T})$	$O(nT^{3/2}\log(T))$	5.3.2
level auction with discretization level m	$O(nm^2\sqrt{T})$	$O(nm^2T)$	5.3.3

Table 5.1: Regret bounds and oracle-based computational efficiency, for the auction classes considered in this work for n bidders and time horizon T. All our results perform a single oracle call per iteration.

such existing routines that are designed to find an optimal auction on historical data, at almost no additional cost. Nevertheless, in some cases one may only be interested in using polynomial time algorithms, even if the solutions they provide are sub-optimal. Therefore, we extend our framework to work with some classes of multiplicative approximation oracles, i.e., oracles that only return an action whose performance is within a constant factor of the performance of the optimal action in class. As a concrete example of the power of such methods, we provide a fully efficient polynomial time online algorithm for the problem of online Welfare Maximization in multi-unit auctions using an offline approximation algorithm. See the extensions for an overview of these results.

5.1.3 Extensions and Additional Applications

In Sections 5.4-5.6, we present several extensions and additional applications of our results. See Table 5.2 for a summary. In addition to below, we refer the interested reader to the full version of our results [113] for additional extensions to contextual online learning platform and extensions to weaker oracle models.

Markovian Adversaries and Competing with the Optimal Auction (Section 5.4). Morgenstern and Roughgarden [210] show that level auctions can provide an arbitrarily accurate approximation to the overall optimal Myerson auction in the Bayesian single-item auction setting if the values of the bidders are drawn from independent distributions and i.i.d. across time. Therefore, if the environment in an online setting picks bidder valuations from independent distributions, standard online-to-batch reductions imply that the revenue of Generalized FTPL with the class of level auctions is close to the overall optimal (i.e., not just best-in-class) single-shot auction. We generalize this reasoning and show the same strong optimality guarantee when the valuations of bidders on each round are drawn from a fast-mixing Markov process that is independent across bidders but Markovian over rounds. For this setting, our results give an oracle-efficient algorithm with regret $O(n^{1/5}T^{9/10})$ to the overall optimal auction, rather than just best-in-class. This is the first result on competing with the Myerson optimal auction for non-i.i.d. distributions, as all prior work [83, 103, 210, 237] assumes i.i.d. samples.

Approximate Oracles and Approximate Regret (Section 5.5). For some problems there might not exist a sufficiently fast (e.g., polynomial-time or FPTAS) offline oracle with small

Problem Class	Regret	Section	Notes
Markovian, single item	$O(n^{1/5}T^{9/10})$	5.4.2	competes with Myerson optimal auction
welfare maximization, s-unit[4]	$1/2$-regret: $O(n^4\sqrt{T})$	5.6.1	fully polynomial-time algorithm
bidding in SiSPAs, k items	$O(km\sqrt{T})$	5.6.2	solves an open problem of [94]

Table 5.2: Additional results considered in Sections 5.4-5.6 and their significance. Above, m is the discretization level of the problems, n is the number of bidders, and T is the time horizon.

additive error as we require. To make our results more applicable in practice, we extend them to handle oracles that are required only to return an action with performance that is within a constant multiplicative factor, $C \leq 1$, of that of the optimal action in the class. We consider two examples of such oracles: Relaxation-based Approximations (see, e.g., [31]) and Maximal-in-Range (MIR) algorithms [216]. Our results hold in both cases with a modified version of regret, called *C-regret*, in which the online algorithm competes with C times the payoff of the optimal action in hindsight.

Additional Applications (Section 5.6). Finally, we provide further applications of our work in the area of online combinatorial optimization with MIR approximate oracles, and in the area of no-regret learning for bid optimization in simultaneous second-price auctions.

- In the first application, we give a polynomial-time learning algorithm for online welfare maximization in multi-unit auctions that achieves $1/2$-regret, by invoking the polynomial-time MIR approximation algorithm of Dobzinski and Nisan [110] as an offline oracle.

- In the second application, we solve an open problem raised in the recent work of Daskalakis and Syrgkanis [94], who offered efficient learning algorithms only for the weaker benchmark of no-envy learning, rather than no-regret learning, in simultaneous second-price auctions, and left as an open question the existence of oracle efficient no-regret algorithms. We show that no-regret learning in simultaneous item auctions is efficiently achievable, assuming access to an optimal bidding oracle against a known distribution of opponents bids (equiv, against a distribution of item prices).

5.2 Generalized FTPL and Oracle-Efficient Online Learning

In this section, we introduce the Generalized Follow-the-Perturbed-Leader (Generalized FTPL) algorithm and describe the conditions under which it efficiently reduces online learning to offline optimization.

As described in Section 5.1.1, we consider the following online learning problem. On each round $t = 1, \ldots, T$, a learner chooses an action x_t from a finite set $\mathcal{X}$, and an adversary chooses an action y_t from a set $\mathcal{Y}$, which is not necessarily finite. The learner then observes y_t and receives a payoff $f(x_t, y_t) \in [0, 1]$, where the function f is fixed and known to the learner. The

[4]The regime of interest in this problem is $s \gg n$. Note that our regret is independent of s in this case.

goal of the learner is to obtain low expected regret with respect to the best action in hindsight, i.e., to minimize

$$\textsc{Regret} \coloneqq \mathbb{E}\left[\max_{x\in\mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right],$$

where the expectation is over the randomness of the learner. An online algorithm is called a *no-regret algorithm* if its regret is sublinear in T, which means that its per-round regret goes to 0 as $T \to \infty$. To simplify exposition, we assume that the adversary is oblivious, i.e., that the sequence $y_1, \ldots, y_T$ is chosen up front without knowledge of the learner's realized actions. Our results generalize to adaptive adversaries using standard techniques [94, 163].

A natural first attempt at an online learning algorithm with oracle access would be one that simply invokes the oracle on the historical data at each round and plays the best action in hindsight. In a stochastic environment in which the adversary's actions are drawn i.i.d. from a fixed distribution on each round, this *Follow-the-Leader* approach achieves a regret of $O(\sqrt{T \log |\mathcal{X}|})$. However, because the algorithm is deterministic, it performs poorly in adversarial environments (see e.g., [53]).

To achieve sublinear regret, we use a common scheme, introduced by Kalai and Vempala [169], and optimize over a perturbed objective at each round. Indeed, our algorithm takes its name from Kalai and Vempala's Follow-the-Perturbed-Leader (FTPL) algorithm. Unlike FTPL, we do not generate a separate independent perturbation for each action, because this creates the two problems mentioned in Section 5.1.1. First, FTPL for unstructured payoffs requires creating $|\mathcal{X}|$ independent random variables, which is intractably large in many applications, including the auction design setting considered here. Second, FTPL yields optimization problems that require a stronger offline optimizer than assumed here. We overcome the first problem by working with perturbations that are not necessarily independent across different actions (prior instances of such an approach were known only for online linear [169] and submodular [155] minimization). We address the second problem by implementing such perturbations with synthetic historical samples of adversary actions; this idea was introduced by Daskalakis and Syrgkanis [94], but they did not provide a method of randomly generating such samples in general learning settings. Thus, our work unifies and extends these previous lines of research.

We create shared randomness among actions in $\mathcal{X}$ by drawing a random vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ of some small size N, with components α_j drawn independently from a dispersed distribution D. The payoff of each of the learner's actions is perturbed by a linear combination of these independent variables, as prescribed by a *perturbation translation matrix* $\boldsymbol{\Gamma}$ of size $|\mathcal{X}| \times N$, with entries in $[0, 1]$. The rows of $\boldsymbol{\Gamma}$, denoted $\boldsymbol{\Gamma}_x$, describe the linear combination for each action x. That is, on each round t, the payoff of each learner action $x \in \mathcal{X}$ is perturbed by $\boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x$, and our Generalized FTPL algorithm outputs an action x that approximately maximizes $\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x$. See Algorithm 5.1 for a full specification. (For non-oblivious adversaries, a fresh random vector $\boldsymbol{\alpha}$ is drawn in each round.)

In the remainder of this section, we analyze the properties of matrix $\boldsymbol{\Gamma}$ that guarantee that Generalized FTPL is no-regret and that its perturbations can be efficiently transformed into synthetic history. Together these properties give rise to efficient reductions of online learning to offline optimization.

Algorithm 5.1: Generalized FTPL

1: Input: non-negative matrix $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}| \times N}$, distribution D, and optimization accuracy parameter ϵ.
2: Draw $\alpha_j \sim D$ for $j = 1, \ldots, N$
3: **for** $t = 1, \ldots, T$ **do**
4: Choose any x_t such that for all $x \in \mathcal{X}$,

$$\sum_{\tau=1}^{t-1} f(x_t, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x_t} \geq \sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x - \epsilon$$

5: Observe y_t and receive payoff $f(x_t, y_t)$
6: **end for**

5.2.1 Regret Analysis

To analyze Generalized FTPL, we first bound its regret by the sum of a *stability* term, a *perturbation* term, and an *error* term in the following lemma. While this approach is standard [169], we include a proof in Appendix A.2 for completeness.

Lemma 5.2.1 (ϵ-FTPL Lemma)**.** *For Generalized FTPL, we have*

$$\text{REGRET} \leq \mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - f(x_t, y_t)\right] + \mathbb{E}\left[\boldsymbol{\alpha} \cdot (\mathbf{\Gamma}_{x_1} - \mathbf{\Gamma}_{x^*})\right] + \epsilon T \tag{5.1}$$

where $x^* = \arg\max_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t)$.

In this lemma, the first term measures the stability of the algorithm, i.e., how often the action changes from round to round. The second term measures the strength of the perturbation, that is, how much the perturbation amount differs between the best action and the initial action. The third term measures the aggregated approximation error in choosing x_t that only approximately optimizes $\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x$.

To bound the stability term, we require that the matrix $\mathbf{\Gamma}$ be *admissible* and the distribution D be *dispersed* in the following sense.

Definition 5.2.2 ((κ, δ)-Admissible Translation Matrix)**.** *A translation matrix* $\mathbf{\Gamma}$ *is* admissible *if its rows are distinct. It is* (κ, δ)-admissible *if it is admissible and also:*

1. *the number of distinct elements within each column is at most* κ*,*
2. *distinct elements within each column differ by at least* δ*.*

Definition 5.2.3 ((ρ, L)-Dispersed Distribution)**.** *A distribution* D *on the real line is* (ρ, L)*-dispersed if for any interval of length* L*, the probability measure placed by* D *on this interval is at most* ρ*.*

In the next lemma, we bound the stability term in Equation (5.1) by showing that with high probability, for all rounds t, we have $x_{t+1} = x_t$. At a high level, since all rows of an admissible matrix $\mathbf{\Gamma}$ are distinct, it suffices to show that the probability that $\mathbf{\Gamma}_{x_{t+1}} \neq \mathbf{\Gamma}_{x_t}$ is small. We prove this for each coordinate $\Gamma_{x_{t+1}j}$ separately, by showing that it is only possible to have $\Gamma_{x_{t+1}j} \neq \Gamma_{x_tj}$ when the random variable α_j falls in a small interval, which happens with only small probability for a sufficiently dispersed distribution D.

Lemma 5.2.4. *Consider Generalized FTPL with a (κ, δ)-admissible matrix $\mathbf{\Gamma}$ with N columns and a $\left(\rho, \frac{1+2\epsilon}{\delta}\right)$-dispersed distribution D. Then, $\mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - f(x_t, y_t)\right] \leq 2TN\kappa\rho$.*

Proof. Fix any $t \leq T$. The bulk of the proof will establish that, with high probability, $\mathbf{\Gamma}_{x_{t+1}} = \mathbf{\Gamma}_{x_t}$, which by admissibility implies that $x_{t+1} = x_t$ and therefore $f(x_{t+1}, y_t) - f(x_t, y_t) = 0$.

Fix any $j \leq N$. We first show that $\Gamma_{x_{t+1}j} = \Gamma_{x_tj}$ with high probability. Let V denote the set of values that appear in the j^{th} column of $\mathbf{\Gamma}$. For any value $v \in V$, let x^v be any action that maximizes the perturbed cumulative payoff among those whose $\mathbf{\Gamma}$ entry in the j^{th} column equals v:

$$x^v \in \operatorname*{arg\,max}_{x \in \mathcal{X}:\, \Gamma_{xj} = v} \left[\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x\right] = \operatorname*{arg\,max}_{x \in \mathcal{X}:\, \Gamma_{xj} = v} \left[\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x - \alpha_j v\right].$$

For any $v, v' \in V$, define

$$\Delta_{vv'} = \left(\sum_{\tau=1}^{t-1} f(x^v, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x^v} - \alpha_j v\right) - \left(\sum_{\tau=1}^{t-1} f(x^{v'}, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x^{v'}} - \alpha_j v'\right).$$

Note that x^v and $\Delta_{vv'}$ are independent of α_j, as we removed the payoff perturbation corresponding to α_j.

If $\Gamma_{x_tj} = v$, then by the ϵ-optimality of x_t on the perturbed cumulative payoff, we have $\alpha_j(v' - v) - \epsilon \leq \Delta_{vv'}$ for all $v' \in V$. Suppose $\Gamma_{x_{t+1}j} = v' \neq v$. Then by the ϵ-optimality of x_{t+1}, we have

$$\sum_{\tau=1}^{t-1} f(x^{v'}, y_\tau) + f(x^{v'}, y_t) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x^{v'}} \geq \sum_{\tau=1}^{t-1} f(x^v, y_\tau) + f(x^v, y_t) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x^v} - \epsilon.$$

Rearranging, we obtain for this same v' that

$$\Delta_{vv'} \leq \alpha_j(v' - v) + f(x^{v'}, y_t) - f(x^v, y_t) + \epsilon \leq \alpha_j(v' - v) + 1 + \epsilon.$$

If $v' > v$, then

$$\alpha_j \geq \frac{\Delta_{vv'} - 1 - \epsilon}{v' - v} \geq \min_{\hat{v} \in V,\, \hat{v} > v} \frac{\Delta_{v\hat{v}} - 1 - \epsilon}{\hat{v} - v}$$

and so $\alpha_j(\overline{v} - v) + 1 + \epsilon \geq \Delta_{v\overline{v}}$ where $\overline{v}$ is the value of $\hat{v}$ minimizing the expression on the right. Thus, in this case we have $-\epsilon \leq \Delta_{v\overline{v}} - \alpha_j(\overline{v} - v) \leq 1 + \epsilon$. Similarly, if $v' < v$, then

$$\alpha_j \leq \frac{\Delta_{vv'} - 1 - \epsilon}{v' - v} \leq \max_{\hat{v} \in V,\, \hat{v} < v} \frac{\Delta_{v\hat{v}} - 1 - \epsilon}{\hat{v} - v}$$

and so $\alpha_j(\underline{v} - v) + 1 + \epsilon \geq \Delta_{v\underline{v}}$ where $\underline{v}$ is the value maximizing the expression on the right. In this case we have $-\epsilon \leq \Delta_{v\underline{v}} - \alpha_j(\underline{v} - v) \leq 1 + \epsilon$. Putting this all together, we have

$$\begin{aligned}
\Pr\left[\Gamma_{x_{t+1}j} \neq \Gamma_{x_t j} \,\middle|\, \alpha_k, k \neq j\right] &\leq \Pr\Bigg[\exists v \in V : \; -\epsilon \leq \Delta_{v\overline{v}} - \alpha_j(\overline{v} - v) \leq 1 + \epsilon \quad \text{or} \\
&\qquad\qquad -\epsilon \leq \Delta_{v\underline{v}} - \alpha_j(\underline{v} - v) \leq 1 + \epsilon \,\Bigg|\, \alpha_k, k \neq j\Bigg] \\
&\leq \sum_{v \in V} \Bigg(\Pr\left[\alpha_j \in \left[\tfrac{\Delta_{v\overline{v}} - 1 - \epsilon}{\overline{v} - v}, \tfrac{\Delta_{v\overline{v}} + \epsilon}{\overline{v} - v}\right] \,\middle|\, \alpha_k, k \neq j\right] \\
&\qquad + \Pr\left[\alpha_j \in \left[\frac{-\Delta_{v\underline{v}} - \epsilon}{v - \underline{v}}, \frac{-\Delta_{v\underline{v}} + 1 + \epsilon}{v - \underline{v}}\right] \,\middle|\, \alpha_k, k \neq j\right]\Bigg) \\
&\leq 2\kappa\rho.
\end{aligned}$$

The last line follows from the fact that $\overline{v} - v \geq \delta$ and $v - \underline{v} \geq \delta$, the fact that D is $\left(\rho, \frac{1+2\epsilon}{\delta}\right)$-dispersed, and a union bound.

Since this bound does not depend on the values of α_k for $k \neq j$, we can remove the conditioning and bound $\Pr[\Gamma_{x_{t+1}j} \neq \Gamma_{x_t j}] \leq 2\kappa\rho$. Taking a union bound over all $j \leq N$, we then have that, by admissibility, $\Pr\left[x_{t+1} \neq x_t\right] = \Pr\left[\mathbf{\Gamma}_{x_{t+1}} \neq \mathbf{\Gamma}_{x_t}\right] \leq 2N\kappa\rho$, which implies the result. □

To bound the regret, it remains to bound the perturbation term in Equation (5.1). This bound is specific to the distribution D. Many distribution families, including (discrete and continuous) uniform, Gaussian, Laplacian, and exponential can lead to a sublinear regret when the variance is set appropriately. Here we present a concrete regret analysis for the case of a uniform distribution:

Theorem 5.2.5. *Let $\mathbf{\Gamma}$ be a (κ, δ)-admissible matrix with N columns and let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \sqrt{\delta/((1+2\epsilon)T\kappa)}$. Then, the regret of Generalized FTPL can be bounded as* $\textsc{Regret} \leq O(N\sqrt{(1+2\epsilon)T\kappa/\delta}) + \epsilon T$. *In general, $\kappa \leq 1/\delta$, so this bound is at most $O\big((N/\delta)\sqrt{(1+2\epsilon)T}\big) + \epsilon T$.*

The proof of this theorem follows immediately from Lemmas 5.2.1 and 5.2.4, setting $\rho = \eta(1+2\epsilon)/\delta = \sqrt{(1+2\epsilon)/(T\kappa\delta)}$.

5.2.2 Oracle-Efficient Online Learning

We now define the offline oracle and oracle-efficient online learning framework more formally. Our oracles are defined for real-weighted datasets. Since many natural offline oracles are iterative optimization algorithms, which are only guaranteed to return an approximate solution in finite time, our definition assumes that the oracle takes the desired precision ϵ as an input. For ease of exposition, we assume that all numerical computations, even those involving real numbers, take $O(1)$ time.

Definition 5.2.6 (Offline Oracle)**.** *An* offline oracle $\textsc{Opt}$ *is any algorithm that receives as input a weighted set of adversary actions $S = \{(w_\ell, y_\ell)\}_{\ell \in \mathcal{L}}$ with $w_\ell \in \mathbb{R}^+$, $y_\ell \in \mathcal{Y}$ and a desired*

Algorithm 5.2: Oracle-Based Generalized FTPL

Input: datasets S_j, $j \in [N]$, that implement a matrix $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}| \times N}$
  distribution D with non-negative support.
  an offline oracle OPT
Draw $\alpha_j \sim D$ for $j = 1, \ldots, N$.
for $t = 1, \ldots, T$ **do**
  For all j, let $\alpha_j S_j$ denote the scaled version of S_j, i.e., $\alpha_j S_j := \{(\alpha_j w, y) : (w, y) \in S_j\}$.
  Set $S = \big\{(1, y_1), \ldots, (1, y_{t-1})\big\} \cup \bigcup_{j \le N} \alpha_j S_j$.
  Play $x_t = \textsc{Opt}\big(S, \frac{1}{\sqrt{T}}\big)$.
  Observe y_t and receive payoff $f(x_t, y_t)$.
end for

precision ϵ, and returns an action $\hat{x} = \textsc{Opt}(S, \epsilon)$ such that

$$\sum_{(w,y) \in S} w f(\hat{x}, y) \ge \max_{x \in \mathcal{X}} \sum_{(w,y) \in S} w f(x, y) - \epsilon.$$

Definition 5.2.7 (Oracle Efficiency)**.** *We say that an online algorithm is* oracle-efficient *with per-round complexity $g(T)$ if its per-round running time is $O(g(T))$ with oracle calls counting $O(1)$.*

We next define a property of a translation matrix $\mathbf{\Gamma}$ which allows us to transform the perturbed objective into a dataset, thus achieving oracle-efficiency of Generalized FTPL:

Definition 5.2.8. *A matrix $\mathbf{\Gamma}$ is* implementable *with complexity M if for each $j \in [N]$ there exists a weighted dataset S_j, with $|S_j| \le M$, such that*

$$\text{for all } x, x' \in \mathcal{X}\text{:} \quad \Gamma_{xj} - \Gamma_{x'j} = \sum_{(w,y) \in S_j} w\big(f(x, y) - f(x', y)\big).$$

In this case, we say that weighted datasets S_j, $j \in [N]$, implement $\mathbf{\Gamma}$ with complexity $\max_{j \in [N]} |S_j|$.

One simple but useful example of implementability is when each column j of $\mathbf{\Gamma}$ specifies the payoffs of every learner action under a particular adversary action $y_j \in \mathcal{Y}$, i.e., $\Gamma_{xj} = f(x, y_j)$ for all x. In this case, $S_j = \{(1, y_j)\}$. Using an implementable $\mathbf{\Gamma}$ gives rise to an oracle-efficient variant of the Generalized FTPL, provided in Algorithm 5.2, in which we explicitly set $\epsilon = 1/\sqrt{T}$. Theorem 5.2.9 shows that the output of this algorithm is equivalent to the output of Generalized FTPL and therefore the same regret guarantees hold. Note the assumption that the perturbations α_j are non-negative. The algorithm can be extended to negative perturbations when both $\mathbf{\Gamma}$ and $-\mathbf{\Gamma}$ are implementable.

Theorem 5.2.9. *If $\mathbf{\Gamma}$ is implementable with complexity M, then Algorithm 5.2 is an oracle-efficient implementation of Algorithm 5.1 with $\epsilon = 1/\sqrt{T}$ and has per-round complexity $O\big(T + NM\big)$.*

Proof. To show that the Oracle-Based FTPL procedure (Algorithm 5.2) implements Generalized FTPL (Algorithm 5.1) with $\epsilon = \frac{1}{\sqrt{T}}$, it suffices to show that at each round t, for any x,

$$\begin{aligned} \sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x \geq \max_{x \in \mathcal{X}} \left[\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x \right] - \epsilon \\ \iff \\ \sum_{(w,y) \in Sj} w f(x, y) \geq \max_{x \in \mathcal{X}} \sum_{(w,y) \in Sj} w f(x, y) - \epsilon, \end{aligned} \tag{5.2}$$

Note that if the above equation holds, then at each timestep, the set of actions that $\textsc{Opt}(S, \epsilon)$ can return legally, i.e., actions whose payoffs are an approximation of the optimal payoff, is exactly the same as the set of actions that the offline optimization step of Algorithm 5.1 can legally play. Clearly, if oracle $\textsc{Opt}$ and Algorithm 5.1 employ the same tie breaking rule, then Algorithm 5.1 and Algorithm 5.2 play the same action at every time step. Even if they do not use the same tie-breaking rule, the guarantee over the payoff of Algorithm 5.2 still holds, as the proof of Theorem 5.2.5 does not rely on using any specific tie-breaking rules. Therefore, the theorem is proved if Equation (5.2) holds.

Let us show that Equation (5.2) is indeed true. For $S = \{(1, y_1), \ldots, (1, y_{t-1})\} \cup \bigcup_{j \leq N} \alpha_j S_j$. Consider any $x, x' \in \mathcal{X}$. Then, from the definition of S and by implementability,

$$\begin{aligned} \sum_{(w,y) \in S} w f(x, y) - \sum_{(w,y) \in S} w f(x', y) &= \sum_{\tau=1}^{t-1} [f(x, y_\tau) - f(x', y_\tau)] + \sum_{j \in [N]} \alpha_j \sum_{(w,y) \in S_j} w \left(f(x, y) - f(x', y) \right) \\ &= \sum_{\tau=1}^{t-1} [f(x, y_\tau) - f(x', y_\tau)] + \sum_{j \in [N]} \alpha_j (\Gamma_{xj} - \Gamma_{x'j}) \\ &= \left(\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x \right) - \left(\sum_{\tau=1}^{t-1} f(x', y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x'} \right), \end{aligned}$$

which immediately yields Equation (5.2).

Also, by implementability, the running time to construct the set S is at most $T + NM$. Since there is only one oracle call per round, we get the per-round complexity of $T + NM$. □

As an immediate corollary, we obtain that the existence of a polynomial-time offline oracle implies the existence of polynomial-time online learner with regret $O(\sqrt{T})$, whenever we have access to an implementable and admissible matrix.

Corollary 5.2.10. *Assume that $\boldsymbol{\Gamma} \in [0, 1]^{|\mathcal{X}| \times N}$ is implementable with complexity M and (κ, δ)-admissible, and there exists an approximate offline oracle $\textsc{Opt}\left(\cdot, \frac{1}{\sqrt{T}}\right)$ which runs in time $\mathrm{poly}(N, M, T)$. Then Algorithm 5.2 with distribution D as defined in Theorem 5.2.5 runs in time $\mathrm{poly}(N, M, T)$ and achieves cumulative regret $O(N\sqrt{T\kappa/\delta})$.*

Alternative Notions of Oracles Multiple other notions of offline optimization oracles may be interesting here. We refer the interested reader to the full version of our results in [113] that include an extensive treatment of integer-weighted oracles and pseudo-polynomial oracles.

5.3 Online Auction Design

In this section, we apply the general techniques developed in Section 5.2 to obtain oracle-efficient no-regret algorithms for several common auction classes.

Consider a mechanism-design setting in which a seller wants to allocate $k \geq 1$ heterogeneous resources to a set of n bidders. The allocation to a bidder i is a subset of $\{1, \ldots, k\}$, which we represent as a vector in $\{0,1\}^k$, and the seller has some feasibility constraints on the allocations across bidders. Each bidder $i \in [n]$ has a combinatorial valuation function $v_i \in \mathcal{V}$, where $\mathcal{V} \subseteq \left(\{0,1\}^k \to [0,1]\right)$. We use $\mathbf{v} \in \mathcal{V}^n$ to denote the vector of valuation functions across all bidders. A special case of the setting is that of multi-item auctions for k heterogeneous items, where each resource is an item and the feasibility constraint simply states that no item is allocated to more than one bidder. Another special case is that of *single-parameter (service-based) environments* in which each resource is a service, e.g., receiving a bundle of items in combinatorial auctions with single minded bidders. Formally, each bidders allocation is in $\{0,1\}$, so we treat this as a setting with $k = 1$ resources where the seller has some constraints on which bidders can receive a service simultaneously. We describe this in more detail in Section 5.3.1.

An auction a takes as input a *bid profile* consisting of reported valuations for each bidder, and returns both the allocation for each bidder i and the price that he is charged. In this work, we only consider *truthful auctions*, where each bidder maximizes his utility by reporting his true valuation, irrespective of what other bidders report. We therefore make the assumption that each bidder reports v_i as their bid and refer to $\mathbf{v}$ not only as the valuation profile, but also as the bid profile throughout the rest of this section. The allocation that the bidder i receives is denoted $\mathbf{q}_i(\mathbf{v}) \in \{0,1\}^k$ and the price that he is charged is $p_i(\mathbf{v})$; we allow sets $\mathbf{q}_i(\mathbf{v})$ to overlap across bidders, and drop the argument $\mathbf{v}$ when it is clear from the context. We consider bidders with quasilinear utilities: the utility of bidder i is $v_i(\mathbf{q}_i(\mathbf{v})) - p_i(\mathbf{v})$. For an auction a with price function $\mathbf{p}(\cdot)$, we denote by $\mathrm{Rev}(a, \mathbf{v})$ the *revenue of the auction* for bid profile $\mathbf{v}$, i.e., $\mathrm{Rev}(a, \mathbf{v}) = \sum_{i \in [n]} p_i(\mathbf{v})$.

For single-parameter service-based environments (a special case of which are single-item auctions), we slightly simplify notation and use $v_i \in [0,1]$ to denote the value of bidder i for being served.

Fixing a class of (truthful) auctions $\mathcal{A}$ and a set of possible valuations $\mathcal{V}$, we consider the problem in which on each round $t = 1, \ldots, T$, a learner chooses an auction $a_t \in \mathcal{A}$ while an adversary chooses a bid profile $\mathbf{v}_t \in \mathcal{V}^n$. The learner then observes $\mathbf{v}_t$ and receives revenue $\mathrm{Rev}(a_t, \mathbf{v}_t)$. The goal of the learner is to obtain low expected regret with respect to the best auction from $\mathcal{A}$ in hindsight. That is, we would like to guarantee that

$$\textsc{Regret} := \mathbb{E}\left[\max_{a \in \mathcal{A}} \sum_{t=1}^{T} \mathrm{Rev}(a, \mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(a_t, \mathbf{v}_t)\right] \leq o(T)\mathrm{poly}(n,k).$$

We require our online algorithm to be oracle-efficient, assuming access to an ϵ-approximate offline optimization oracle that takes as input a weighted set of bid profiles, $S = \{(w_\ell, \mathbf{v}_\ell)\}_{\ell \in \mathcal{L}}$ and returns an auction that achieves an approximately optimal revenue on S, i.e., a revenue at least $\max_{a \in \mathcal{A}} \sum_{(w,\mathbf{v}) \in S} w\mathrm{Rev}(a, \mathbf{v}) - \epsilon$. Throughout the section, we assume that there exists such an oracle for $\epsilon = 1/\sqrt{T}$, as needed in Algorithm 5.2.

Using the language of oracle-based online learning developed in Section 5.2, the learner's action corresponds to the choice of auction, the adversary's action corresponds to the choice of bid profile, the payoff of the learner corresponds to the revenue generated by the auction, and we assume access to an offline optimization oracle OPT. These correspondences are summarized in the following table.

Auction Setting	**Oracle-Based Learning Equivalent**
Auctions $a_t \in \mathcal{A}$	Learner actions $x_t \in \mathcal{X}$
Bid/valuation profiles $\mathbf{v}_t \in \mathcal{V}^n$	Adversary actions $y_t \in \mathcal{Y}$
Revenue function Rev	Payoff function f

For several of the auction classes we consider, such as multi-item or multi-unit auctions, the revenue of an auction on a bid profile is in range $[0, R]$ for $R > 1$. In order to use the results of Section 5.2, we implicitly re-scale all the revenue functions by dividing them by R before applying Theorem 5.2.5. Note that, since $\mathbf{\Gamma}$ does not change, the admissibility condition keeps the regret of the normalized problem at $O(N\sqrt{T\kappa/\delta})$, according to Theorem 5.2.5. We then scale up to get a regret bound that is R times the regret for the normalized problem, i.e., $O(RN\sqrt{T\kappa/\delta})$. Assuming that numerical computations take $O(1)$ time, this re-scaling does not increase the runtime, as when both the revenues are scaled down by a factor of R and the matrix $\mathbf{\Gamma}$ is unchanged, the implementability dataset S_j is scaled up by a factor of R which does not change the complexity M of implementing $\mathbf{\Gamma}$. Refer to full version of these results [113] for a note on numerical computations and the mild change in runtime when numerical computations do not take $O(1)$ time.

We now derive results for VCG auctions with bidder-specific reserves, envy-free item-pricing auctions, and level auctions. We defer the definition of each auction class to its respective subsection.

5.3.1 VCG with Bidder-Specific Reserves

In this section, we consider a standard class of auctions, VCG auctions with bidder-specific reserve prices, which we define more formally below and denote by $\mathcal{I}$. These auctions are known to approximately maximize the revenue when the bidder valuations are drawn independently (but not necessarily identically) from some distribution [152]. Recently, Roughgarden and Wang [239] considered this class $\mathcal{I}$ in an online learning framework. They provided a computationally efficient algorithm whose total revenue is at least $1/2$ of the best revenue among auctions in $\mathcal{I}$, minus a term that is $o(T)$. We apply the techniques from Section 5.2 to generate an oracle-efficient online algorithm with low *additive* regret with respect to the optimal auction in the class $\mathcal{I}$, without any loss in multiplicative factors.

We go beyond single-item auctions and consider general *single-parameter* environments. In these environments, each bidder has one piece of private valuation for receiving a *service*, i.e., being included in the set of winning bidders. We allow for some combinations of bidders to be *served* simultaneously, and let $\mathcal{S} \subseteq 2^{[n]}$ be the family of feasible sets, i.e., sets of bidders that can be served simultaneously; with some abuse of notation we write $\mathbf{q} \in \mathcal{S}$, to mean that the set represented by the binary allocation vector $\mathbf{q}$ is in $\mathcal{S}$. We assume that it is possible for any bidder

to be the sole bidder served, i.e., that $\{i\} \in \mathcal{S}$ for all i, and that it is possible that no bidder is served, i.e., $\emptyset \in \mathcal{S}$.[5] Examples of such environments include single-item single-unit auctions (for which $\mathcal{S}$ contains only singletons and the empty set), single-item s-unit auctions (for which $\mathcal{S}$ contains any subset of size at most s), and combinatorial auctions with single-minded bidders. In the last case, we begin with some set of original items, define the service as receiving the desired bundle of items, and let $\mathcal{S}$ contain any subset of bidders seeking disjoint sets of items.

We consider the class of VCG auctions with bidder-specific reserves. In a basic VCG auction, an allocation $\mathbf{q}^* \in \mathcal{S}$ is chosen to maximize social welfare, that is, maximize $\sum_{i=1}^{n} v_i q_i^*$. Each bidder who is served is then charged the externality he imposes on others, $p_i(\mathbf{v}) = \max_{\mathbf{q} \in \mathcal{S}} \sum_{i' \neq i} v_{i'} q_{i'} - \sum_{i' \neq i} v_{i'} q_{i'}^*$, which can be shown to equal the minimum bid at which he would be served. Such auctions are known to be truthful. The most common example is the second-price auction for the single-item single-unit case in which the bidder with the highest bid receives the item and pays the second highest bid. VCG auctions with reserves, which maintain the property of truthfulness, are defined as follows.

Definition 5.3.1 (VCG auctions with bidder-specific reserves)**.** *A VCG auction with bidder-specific reserves is specified by a vector* $\mathbf{r}$ *of reserve prices for each bidder. As a first step, all bidders whose bids are below their reserves (that is, bidders* i *for which* $v_i < r_i$*) are removed from the auction. If no bidders remain, no item is allocated. Otherwise, the basic VCG auction is run on the remaining bidders to determine the allocation. Each bidder who is served is charged the larger of his reserve and his VCG payment.*

Fixing the set $\mathcal{S}$ of feasible allocations, we denote by $\mathcal{I}$ the class of all VCG auctions with bidder-specific reserves. With a slight abuse of notation we write $\mathbf{r} \in \mathcal{I}$ to denote the auction with reserve prices $\mathbf{r}$. To apply the results from Section 5.2, which require a finite action set for the learner, we limit attention to the finite set of auctions $\mathcal{I}_m \subseteq \mathcal{I}$ consisting of those auctions in which the reserve price for each bidder is a strictly positive integer multiple of $1/m$, i.e., those where $r_i \in \{1/m, \ldots, m/m\}$ for all i. We will show for some common choices of $\mathcal{S}$ that the best auction in this class yields almost as high revenue as the best auction in $\mathcal{I}$.

We next show how to design a matrix $\mathbf{\Gamma}$ for this problem that is admissible and implementable. As a warmup, suppose we use the $|\mathcal{I}_m| \times n$ matrix $\mathbf{\Gamma}$ with $\Gamma_{\mathbf{r}i} = \mathrm{Rev}(\mathbf{r}, \mathbf{e}_i)$ for all $\mathbf{r} \in \mathcal{I}_m$ and $i \in [n]$. That is, the i^{th} column of $\mathbf{\Gamma}$ corresponds to the revenue of each auction on a bid profile in which bidder i has valuation 1 and all others have valuation 0. By definition, $\mathbf{\Gamma}$ is implementable with complexity n using $S_j = \{(1, \mathbf{e}_j)\}_{j \in [n]}$. Moreover, $\mathrm{Rev}(\mathbf{r}, \mathbf{e}_i) = r_i$ so any two rows of $\mathbf{\Gamma}$ are indeed different and $\mathbf{\Gamma}$ is $(m, 1/m)$-admissible. By Theorem 5.2.9, there is an oracle-efficient implementation of the Generalized FTPL with regret that is polynomial in m.

To improve this regret bound and obtain a regret that is polynomial in $\log(m)$, we carefully construct another translation matrix that is implementable using a more complex dataset of adversarial actions. As we describe shortly, the translation matrix we design is quite intuitive. The row corresponding to an auction $\mathbf{r}$ contains a binary representation of its reserve prices. In this case, proving admissibility of the matrix is simple. The challenge is then showing that this simple translation matrix is implementable using a dataset of adversarial actions.

[5] A more common and stronger assumption used in previous work [152, 239] is that $\mathcal{S}$ is a downward closed matroid.

Construction of Γ: Let Γ^{VCG} be an $|\mathcal{I}_m| \times (n\lceil \log m \rceil)$ binary matrix, where the i^{th} collection of $\lceil \log m \rceil$ columns contain the binary encodings of the auctions' reserve prices for bidder i. More formally, for any $i \leq n$ and a bit position $\beta \leq \lceil \log m \rceil$, let $j = (i-1)\lceil \log m \rceil + \beta$ and set $\Gamma^{\text{VCG}}_{\mathbf{r}j}$ to be the β^{th} bit of mr_i.

Lemma 5.3.2. *Γ^{VCG} is $(2,1)$-admissible and implementable with complexity m.*

Let us first illustrate the main ideas used in the proof of Lemma 5.3.2 through a simple example.

	Binary encoding			
Auction $\mathbf{r}$	r_1		r_2	
$(1/3, 1/3)$	0	1	0	1
$(1/3, 2/3)$	0	1	1	0
$(1/3, 3/3)$	0	1	1	1
$(2/3, 1/3)$	1	0	0	1
$(2/3, 2/3)$	1	0	1	0
$(2/3, 3/3)$	1	0	1	1
$(3/3, 1/3)$	1	1	0	1
$(3/3, 2/3)$	1	1	1	0
$(3/3, 3/3)$	1	1	1	1

$\Delta = -1$

$\Delta = 0$

$\Delta' = 1$

$\Delta' = -1$

Figure 5.1: Γ^{VCG} for $n = 2$ bidders and $m = 3$

Example 5.3.3. *Consider Γ^{VCG} for $n = 2$ bidders and $m = 3$ discretization levels, as demonstrated in Figure 5.1. As an example, we show how one can go about implementing columns 1 and 4 of Γ^{VCG}.*

Consider the first column of Γ^{VCG}. This corresponds to the most significant bit of r_1, so this value is independent of the value of r_2. Hence, to implement this column, we need to find a set of bid profiles where the difference in revenue is as prescribed by Γ^{VCG}. Consider bid profiles $\mathbf{v}_h = (h/3, 0)$ for $h \in \{1,2,3\}$. This requirement is satisfied by $S_1 = \{(w_h, \mathbf{v}_h)\}_h$ iff the weights satisfy the following two equations:

$$\frac{1}{3}(w_1 + w_2 + w_3) - \frac{2}{3}(w_2 + w_3) = -1,$$
$$\frac{2}{3}(w_2 + w_3) - \frac{3}{3}(w_3) = 0,$$

where the left-hand sides of the two equations are the differences in the revenues of two reserve prices $r_1 = \frac{1}{3}$ and $\frac{2}{3}$, and $r_1 = \frac{2}{3}$ and $\frac{3}{3}$, respectively, and the right-hand sides are the differences between the corresponding entries of Γ^{VCG} (denoted by Δ in Figure 5.1). Note that $S_1 = \{(3, \mathbf{v}_1), (2, \mathbf{v}_2), (4, \mathbf{v}_3)\}$ satisfies this requirement and implements the first column. Similarly, for implementing the fourth column we consider bid profiles $\mathbf{v}'_h = (0, h/3)$ for $h \in \{1,2,3\}$ and equations dictated by the values of Γ^{VCG} and values Δ'. One can verify that $S_4 = \{(6, \mathbf{v}'_1), (0, \mathbf{v}'_2), (3, \mathbf{v}'_3)\}$ implements this column.

More generally, the proof of Lemma 5.3.2 shows that Γ^{VCG} is implementable by showing that any differences in values in one column that solely depend on a single bidder's reserve price lead to a system of linear equations that is satisfiable.

Let us now turn our attention to the proof of Lemma 5.3.2.

Proof of Lemma 5.3.2. In the interest of readability, we drop the superscript and write $\mathbf{\Gamma}$ for $\mathbf{\Gamma}^{\text{VCG}}$ in this proof.

For any $\mathbf{r}$, row $\mathbf{\Gamma}_{\mathbf{r}}$ corresponds to the binary encoding of $r_1, \ldots, r_n$. Therefore, for any two different auctions $\mathbf{r} \neq \mathbf{r}'$, $\mathbf{\Gamma}_{\mathbf{r}} \neq \mathbf{\Gamma}_{\mathbf{r}'}$. Since $\mathbf{\Gamma}$ is a binary matrix, this implies that $\mathbf{\Gamma}$ is $(2,1)$-admissible.

Next, we prove that $\mathbf{\Gamma}$ is implementable. Pick $i \leq n$ and $\beta \leq \lceil \log m \rceil$, and the associated column index j. We will construct the set S_j for each column $j \leq n\lceil \log(m) \rceil$ for implementing $\mathbf{\Gamma}$. The set S_j includes exactly the m profiles in which only the bidder i has non-zero valuation, denoted as $\mathbf{v}_h \coloneqq (h/m)\mathbf{e}_i$ for $h \leq m$. To determine their weights w_h, we use the definition of implementability. In particular, the weights must satisfy:

$$\forall\, \mathbf{r}, \mathbf{r}' \in \mathcal{I}_m, \qquad \Gamma_{\mathbf{r}j} - \Gamma_{\mathbf{r}'j} = \sum_{h \leq m} w_h \Big(\text{Rev}(\mathbf{r}, \mathbf{v}_h) - \text{Rev}(\mathbf{r}', \mathbf{v}_h) \Big).$$

In the above equation, $\Gamma_{\mathbf{r}j}$ and $\Gamma_{\mathbf{r}'j}$ encode the β^{th} bit of r_i and r_i', respectively, so the left-hand side is independent of the reserve prices for bidders $i' \neq i$. Moreover, $\text{Rev}(\mathbf{r}, \mathbf{v}_h) = r_i \mathbf{1}_{(h \geq m r_i)}$, so the right-hand side of the above equation is also independent of the reserve prices for bidders $i' \neq i$. Let z_β be the β^{th} bit of integer z. That is, $\Gamma_{\mathbf{r}j} = (m r_i)_\beta$. Substituting $z = m r_i$ and $z' = m r_i'$, the above equation can be reformulated as

$$\forall z, z' \in \{1, \ldots, m\}, \quad \big(z_\beta - z'_\beta\big) = \sum_{h \leq m} w_h \left(\frac{z}{m} \mathbf{1}_{(h \geq z)} - \frac{z'}{m} \mathbf{1}_{(h \geq z')} \right).^{6} \tag{5.3}$$

We next recursively derive the weights w_h, and show that they are non-negative and satisfy Equation (5.3). To begin, let

$$w_m = \max\Big\{0,\ \max_z \big\lceil m\big(z_\beta - (z-1)_\beta\big)\big\rceil \Big\},$$

and for all $z = m, m-1, \ldots, 2$, define

$$w_{z-1} = \frac{1}{z-1} \left(\sum_{h=z}^{m} w_h - m\big(z_\beta - (z-1)_\beta\big) \right).$$

Next, we show by induction that $w_h \geq 0$ for all h. For the base case of $h = m$, by definition $w_m \geq 0$. Now, assume that for all $h \geq z$, $w_h \geq 0$. Then

$$w_{z-1} \geq \frac{1}{z-1} \Big(w_m - m\big(z_\beta - (z-1)_\beta\big) \Big) \geq 0.$$

Therefore all weights are non-negative. Furthermore, by rearranging the definition of w_{z-1}, we have

$$\begin{aligned} \big(z_\beta - (z-1)_\beta\big) &= \frac{1}{m} \left(\sum_{h=z}^{m} w_h - (z-1) w_{z-1} \right) = \frac{1}{m} \left(z \sum_{h=z}^{m} w_h - (z-1) \sum_{h=z-1}^{m} w_h \right) \\ &= \sum_{h \leq m} w_h \left(\frac{z}{m} \mathbf{1}_{(h \geq z)} - \frac{z-1}{m} \mathbf{1}_{(h \geq z-1)} \right). \end{aligned}$$

[6] Not including the reserve 0 is a crucial technical point for the proof of implementability.

Where in the second equality we simply added and subtracted the term $(z-1)\sum_{h=z}^{m} w_h$ and in the last equality, we grouped together common terms.

Equation (5.3) is proved for a particular pair $z > z'$ by summing the above expression for $\left(\zeta_\beta - (\zeta-1)_\beta\right)$ over all $\zeta \in (z', z]$ and canceling telescoping terms, and if $z = z'$, the statement holds regardless of the weights chosen.

This shows that $\mathbf{\Gamma}$ is implementable. Note that the cardinality of S_j is at most m. Also note that the above proof constructs (in $\mathrm{poly}(n, m)$ time) such datasets $\{S_j\}_{j\in[n\lceil\log m\rceil]}$ that implement $\mathbf{\Gamma}$. Therefore, $\mathbf{\Gamma}$ is implementable with complexity m using the datasets described above.

□

The next theorem follows immediately from Lemma 5.3.2, Theorem 5.2.5, and the fact that the maximum revenue is at most R.

Theorem 5.3.4. *Consider the online auction design problem for the class of VCG auctions with bidder-specific reserves, $\mathcal{I}_m$. Let $R = \max_{\mathbf{r},\mathbf{v}} \mathrm{Rev}(\mathbf{r}, \mathbf{v})$ and let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\mathrm{VCG}}$ is oracle-efficient with per-round complexity $\mathrm{poly}(n, m, T)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{r}\in\mathcal{I}_m}\sum_{t=1}^{T}\mathrm{Rev}(\mathbf{r},\mathbf{v}_t) - \sum_{t=1}^{T}\mathrm{Rev}(\mathbf{r}_t,\mathbf{v}_t)\right] \le O(n\log(m)R\sqrt{T}).$$

Note that R is bounded by the number of bidders that can be served simultaneously, which is at most n.

Now we return to the infinite class $\mathcal{I}$ of all VCG auctions with reserve prices $r_i \in [0, 1]$. We show $\mathcal{I}_m$ is a finite "cover" for this class when the family of feasible sets $\mathcal{S}$ is the set of all subsets of size at most s, corresponding to single-item single-unit auctions (when $s = 1$) or more general single-item s-unit auctions. In particular, we prove in Appendix A.1, that the optimal revenue of $\mathcal{I}_m$ compared with that of $\mathcal{I}$ can decrease by at most $2s/m$ at each round. That is,

$$\max_{\mathbf{r}\in\mathcal{I}}\sum_{t=1}^{T}\mathrm{Rev}(\mathbf{r},\mathbf{v}_t) - \max_{\mathbf{r}\in\mathcal{I}_m}\sum_{t=1}^{T}\mathrm{Rev}(\mathbf{r},\mathbf{v}_t) \le \frac{2Ts}{m}. \tag{5.4}$$

Setting $m = \sqrt{T}$ and using Theorem 5.3.4, we obtain the following result for the class of auctions $\mathcal{I}$.

Theorem 5.3.5. *Consider the online auction design problem for the class of VCG auctions with bidder-specific reserves, $\mathcal{I}$, in s-unit auctions. Let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\mathrm{VCG}}$ is oracle-efficient with per-round complexity $\mathrm{poly}(n, T)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{r}\in\mathcal{I}}\sum_{t=1}^{T}\mathrm{Rev}(\mathbf{r},\mathbf{v}_t) - \sum_{t=1}^{T}\mathrm{Rev}(\mathbf{r}_t,\mathbf{v}_t)\right] \le O(ns\log(T)\sqrt{T}).$$

5.3.2 Envy-free Item Pricing

In this section, we consider envy-free item pricing [141] in an environment with k heterogeneous items with a supply of $s_\ell \geq 0$ units for each item $\ell \leq k$.

Definition 5.3.6 (Envy-free Item-Pricing Auction)**.** *An* envy-free item-pricing auction *for k heterogeneous items, given supply s_ℓ for $\ell = 1, \ldots, k$, is defined by a vector of prices $\mathbf{a}$, where a_ℓ is the price of item ℓ. The mechanism considers bidders $i = 1, \ldots, n$ in order and allocates to bidder i the bundle $\mathbf{q}_i \in \{0,1\}^k$ that maximizes $v_i(\mathbf{q}_i) - \mathbf{a} \cdot \mathbf{q}_i$, among all feasible bundles, i.e., bundles that can be composed from the remaining supplies. Bidder i is then charged the price $\mathbf{a} \cdot \mathbf{q}_i$.*

Examples of such environments include *unit-demand* bidders and *single-minded* bidders in settings such as *hypergraph pricing*, where bidders seek hyper-edges in a hypergraph, and its variant *the highway problem*, where bidders seek hyperedges between sets of contiguous vertices [31, 141]. We describe some of these problems in more detail later on.

We represent by $\mathcal{P}_m$ the class of all such envy-free item pricing auctions where all the prices are strictly positive multiples of $1/m$, i.e., $a_\ell \in \{1/m, \ldots, m/m\}$ for all ℓ. Next, we discuss the construction of an implementable and admissible translation matrix $\mathbf{\Gamma}$. Consider a bid profile where one bidder has value v for bundle $\mathbf{e}_\ell$ and all other bidders have value 0 for all bundles. The revenue of auction $\mathbf{a}$ on such a bid profile is $a_\ell \mathbf{1}_{(v \geq a_\ell)}$. Note the similarity to the case of VCG auctions with bidder-specific reserve prices $\mathbf{r}$, where bid profiles with a single non-zero valuation v_i and revenue $r_i \mathbf{1}_{(v_i \geq r_i)}$ were used to create an implementable construction for $\mathbf{\Gamma}$. We show that a similar construction works for $\mathcal{P}_m$.

Construction of $\mathbf{\Gamma}$: Let $\mathbf{\Gamma}^{\text{IP}}$ be a $|\mathcal{P}_m| \times (k\lceil \log m \rceil)$ binary matrix, where the ℓ^{th} collection of $\lceil \log m \rceil$ columns correspond to the binary encoding of the auction's price for item ℓ. More formally, for any $\ell \leq k$ and $\beta \leq \lceil \log m \rceil$, $\Gamma^{\text{IP}}_{\mathbf{a}j}$ is the β^{th} bit of (the integer) ma_ℓ, where $j = (\ell - 1)\lceil \log m \rceil + \beta$. Next, we show that $\mathbf{\Gamma}^{\text{IP}}$ is admissible and implementable. The proof of the following lemma is analogous to that of Lemma 5.3.2.

Lemma 5.3.7. *$\mathbf{\Gamma}^{\text{IP}}$ is $(2,1)$-admissible and implementable with complexity m.*

Proof. We will argue that the setting here is isomorphic to the setting in the proof of Lemma 5.3.2, so we can directly apply the result of analysis of $\mathbf{\Gamma}^{\text{VCG}}$. The isomorphism from the VCG setting to IP setting maps bidders i in VCG to items ℓ in IP, and reserve price vectors $\mathbf{r}$ to price vectors $\mathbf{a}$. We therefore assume that n in VCG equals k in IP, and the values of m in both settings are equal. Then, indeed $\mathbf{\Gamma}^{\text{VCG}}$ equals $\mathbf{\Gamma}^{\text{IP}}$.

Next we need to show how to construct S_j for all j in the $\mathbf{\Gamma}^{\text{IP}}$ setting. Assume that j corresponds to the bidder i and the bit β in VCG setting, and the item ℓ and the bit β in IP setting. In VCG, we considered the bid profiles $\mathbf{v}_h = (h/m)\mathbf{e}_i$, and the revenue of any auction $\mathbf{r}$ is

$$\text{Rev}^{\text{VCG}}(\mathbf{r}, \mathbf{v}_h) = r_i \mathbf{1}_{(h \geq m r_i)}.$$

In IP setting, we consider profiles $\mathbf{v}'_h$ of combinatorial valuations over bundles $\mathbf{q} \in \{0,1\}^k$, in which all bidders have values zero on all bundles and one bidder has value h/m for bundle $\mathbf{e}_\ell$

and zero on all other bundles.[7] In this case, we have

$$\mathrm{Rev}^{\mathrm{IP}}(\mathbf{a}, \mathbf{v}'_h) = a_i \mathbf{1}_{(h \geq m a_i)}.$$

Thus, both the translation matrices $\mathbf{\Gamma}^{\mathrm{VCG}}$ and $\mathbf{\Gamma}^{\mathrm{IP}}$ as well as the revenue functions $\mathrm{Rev}^{\mathrm{VCG}}$ and $\mathrm{Rev}^{\mathrm{IP}}$ are isomorphic (given these choices of the profiles). Therefore, we can set the weights w'_h in IP setting equal to the weights w_h in VCG setting and obtain admissibility and implementability with the same constants and complexity. □

Our main theorem follows immediately from Lemma 5.3.7, Theorems 5.2.5 and 5.2.9, and the fact that the revenue of the mechanism at every step is at most R. In general, R is at most n.

Theorem 5.3.8. *Consider the online auction design problem for the class of envy-free item pricing auctions, $\mathcal{P}_m$. Let $R = \max_{\mathbf{a},\mathbf{v}} \mathrm{Rev}(\mathbf{a}, \mathbf{v})$ and let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\mathrm{IP}}$ is oracle-efficient with per-round complexity $\mathrm{poly}(k, m, T)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{a}\in\mathcal{P}_m} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{a}, \mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{a}_t, \mathbf{v}_t)\right] \leq O\left(kR \log(m)\sqrt{T}\right).$$

Consider the class of all envy-free item-pricing auctions where $a_\ell \in [0, 1]$ is a real number and denote this class by $\mathcal{P}$. We show that $\mathcal{P}_m$ is a discrete "cover" for $\mathcal{P}$ when there is an *unlimited supply* of all items ($s_\ell = \infty$ for all ℓ) and the bidders have *single-minded* or *unit-demand* valuations. In the single-minded setting, each bidder i is interested in one particular bundle of items $\hat{\mathbf{q}}_i$. That is, $v_i(\mathbf{q}_i) = v_i(\hat{\mathbf{q}}_i)$ for all $\mathbf{q}_i \supseteq \hat{\mathbf{q}}_i$ and 0 otherwise. In the unit-demand setting, each bidder i has valuation $v_i(\mathbf{e}_\ell)$ for item ℓ, and wishes to purchase *at most one item*, i.e., item $\arg\max_\ell (v_i(\mathbf{e}_\ell) - a_\ell)$. We show that in both settings, discretizing item prices cannot decrease revenue by much (see Appendix A.3).

Lemma 5.3.9. *For any $\mathbf{a} \in \mathcal{P}$ there is $\mathbf{a}' \in \mathcal{P}_m$, such that for any unit-demand valuation profile or for any single-minded valuation profile $\mathbf{v}$ with infinite supply (the digital goods setting), $\mathrm{Rev}(\mathbf{a}, \mathbf{v}) - \mathrm{Rev}(\mathbf{a}', \mathbf{v}) \leq 2nk/m$.*

These discretization arguments together with Theorem 5.3.8 yield the following result for the class of auctions $\mathcal{P}$ (using the fact that $R \leq n$, and the setting $m = \sqrt{T}$):

Theorem 5.3.10. *Consider the online auction design problem for the class of envy-free item pricing auctions, $\mathcal{P}$, with unit-demand bidders or with single-minded bidders with infinite supply (the digital goods setting). Let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\mathrm{IP}}$ is oracle-efficient with per-round complexity $\mathrm{poly}(k, T)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{a}\in\mathcal{P}} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{a}, \mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{a}_t, \mathbf{v}_t)\right] \leq O\left(nk \log(T)\sqrt{T}\right).$$

[7] Note that a simple variation of this bid profile can be used in settings where the valuations need to satisfy additional assumptions, such as (sub-)additivity or free disposal. In such cases, we can use a similar bid profile where one bidder has valuation h/m for any bundle that includes item ℓ and all other valuations are 0.

5.3.3 Level Auctions

We next consider the class of *level auctions* introduced by Morgenstern and Roughgarden [210]. These auctions can achieve $(1-\epsilon)$-approximate revenue maximization if the valuations of the bidders are drawn independently (but not necessarily identically) from a distribution [210], approximating Myerson's optimal auction [212]. Using our tools, we derive oracle-efficient no-regret algorithms for this auction class.

The s-level auctions realize a single-item single-unit allocation as follows.

Definition 5.3.11. *An s-level auction $\boldsymbol{\theta}$ is defined by s thresholds for each bidder i, $0 \le \theta_0^i < \cdots < \theta_{s-1}^i \le 1$. For any bid profile $\mathbf{v}$, we let $b_i^{\boldsymbol{\theta}}(v_i)$ denote the index b of the largest threshold $\theta_b^i \le v_i$, or -1 if $v_i < \theta_0^i$. If $v_i < \theta_0^i$ for all i, the item is not allocated. Otherwise, the item goes to the bidder with the largest index $b_i^{\boldsymbol{\theta}}(v_i)$, breaking ties in favor of bidders with smaller i. The winner pays the price equal to the minimum bid that he could have submitted and still won the item.*

When it is clear from the context, we omit $\boldsymbol{\theta}$ in $b_i^{\boldsymbol{\theta}}(v_i)$ and write just $b_i(v_i)$. We consider a class of s-level auctions, $\mathcal{S}_{s,m}$ that is the set of all auctions described by Definition 5.3.11 with thresholds that are in the set $\{0, \frac{1}{m}, \ldots, \frac{m}{m}\}$.

Let us discuss a construction of an admissible and implementable $\boldsymbol{\Gamma}$ for $\mathcal{S}_{s,m}$. Our approach for designing matrix $\boldsymbol{\Gamma}$ starts with a matrix that is clearly implementable, but the challenge is in showing that it is also admissible. In what follows, we identify a small subset of the actions of the adversary, such that any two actions of the learner receive sufficiently different revenues on at least one of these actions. This naturally leads to an admissible and implementable construction for $\boldsymbol{\Gamma}$.

Consider the bid profile in which the only non-zero bids are $v_n = \ell/m$ for some $0 \le \ell \le m$, and $v_i = 1$ for a single bidder $i < n$. Note that bidder i wins the item in any such profile and pays θ_b^i corresponding to $b = \max\{0, b_n(v_n)\}$. We define a matrix $\boldsymbol{\Gamma}$ with one column for every bid profile of this form and an additional column for the bid profile $\mathbf{e}_n$, with the entries in each row consisting of the revenue of the corresponding auction on the given bid profile. Clearly, $\boldsymbol{\Gamma}$ is implementable. As for admissibility, take $\boldsymbol{\theta} \in \mathcal{S}_{s,m}$ and the corresponding row $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$. Note that as $v_n = \ell/m$ increases for $\ell = 0, \ldots, m$, there is an increase in $b_n(\ell/m) = -1, 0, \ldots, s-1$, possibly skipping the initial -1. As the level $b_n(v_n)$ increases, the auction revenue attains the values $\theta_0^i, \theta_1^i, \ldots, \theta_{s-1}^i$, changing exactly at those points where v_n crosses thresholds $\theta_1^n, \ldots, \theta_{s-1}^n$. Since any two consecutive thresholds of $\boldsymbol{\theta}$ are different, the thresholds of θ_b^i for $b \ge 0$ and θ_b^n for $b \ge 1$ can be reconstructed by analyzing the revenue of the auction and the values of v_n at which the revenue changes. The remaining threshold θ_0^n can be recovered by examining the revenue of the bid profile $\mathbf{v} = \mathbf{e}_n$. Since all of the parameters of the auction can be recovered from the entries in the row, this shows that any two rows of $\boldsymbol{\Gamma}$ are different and $\boldsymbol{\Gamma}$ is $(m+1,\, 1/m)$-admissible. This reasoning is summarized in the following construction and the corresponding lemma. See Figure 5.2 for more intuition.

Construction of $\boldsymbol{\Gamma}$: For $i \in \{1, \ldots, n-1\}$ and $\ell \in \{0, \ldots, m\}$, let $\mathbf{v}^{i,\ell} = \mathbf{e}_i + (\ell/m)\mathbf{e}_n$. Let $V = \{\mathbf{v}^{i,\ell}\}_{i,\ell} \cup \{\mathbf{e}_n\}$. Let $\boldsymbol{\Gamma}^{\mathrm{SL}}$ be the matrix of size $|\mathcal{S}_{s,m}| \times |V|$ with entries indexed by $(\boldsymbol{\theta}, \mathbf{v}) \in \mathcal{S}_{s,m} \times V$, such that $\Gamma^{\mathrm{SL}}_{\boldsymbol{\theta},\mathbf{v}} = \mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})$.

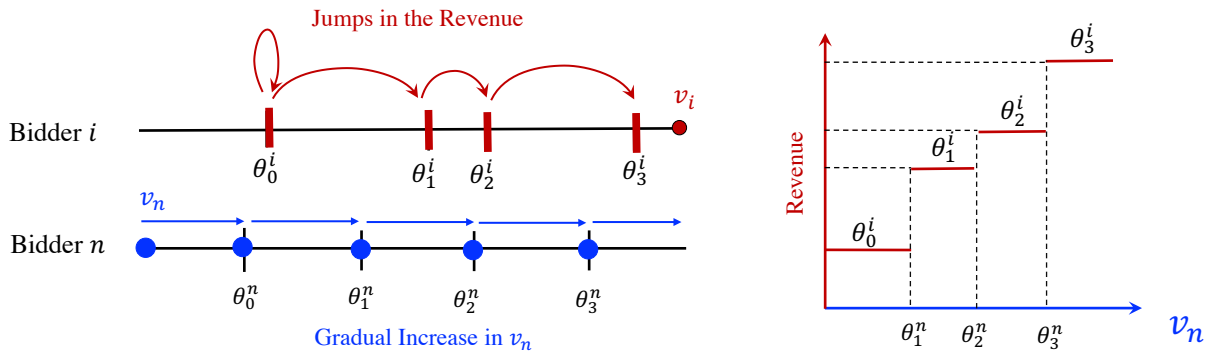
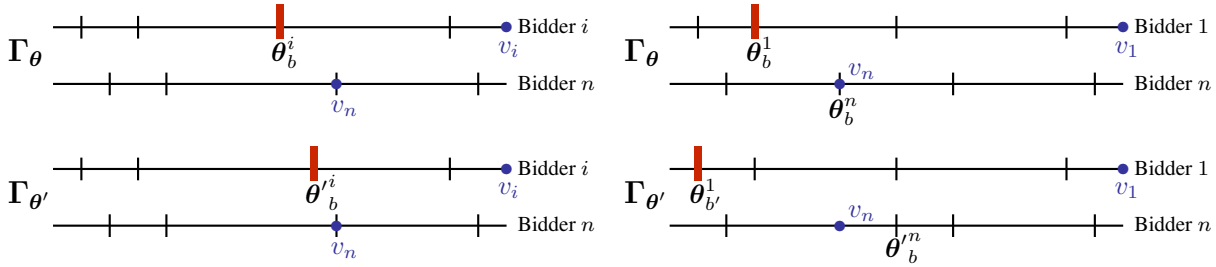


Figure 5.2: Demonstration of how $\boldsymbol{\theta}$ can be reconstructed by its revenue on the bid profiles in $V = \{\mathbf{v}^{i,\ell}\}_{i,\ell} \cup \{\mathbf{e}_n\}$. On the left, we show that as the value v_n (blue circle) gradually increases from 0 to 1, the revenue of the auction (red vertical lines) jumps along the sequence of values $\theta_0^i, \theta_1^i, \ldots, \theta_{s-1}^i$. So by analyzing the revenue of an auction on all bid profiles $\{\mathbf{v}^{i,\ell}\}_{i,\ell}$ one can reconstruct $\boldsymbol{\theta}^i$ for $i \neq n$ and $\theta_1^n, \ldots, \theta_{s-1}^n$. To reconstruct θ_0^n, one only needs to consider the profile $\mathbf{e}_n$. The figure on the right demonstrates the revenue of the same auction, where the horizontal axis is the value of v_n and the vertical axis is the revenue of the auction when $v_i = 1$ and all other valuations are 0.

Lemma 5.3.12. $\boldsymbol{\Gamma}^{\mathrm{SL}}$ *is* $(m+1, 1/m)$*-admissible and implementable with complexity* 1.

Proof. Since $\Gamma^{\mathrm{SL}}_{\boldsymbol{\theta},\mathbf{v}} = \mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})$, $\boldsymbol{\Gamma}^{\mathrm{SL}}$ can be implemented by datasets $S_{\mathbf{v}} = \{(1, \mathbf{v})\}$ for $\mathbf{v} \in V$. So, $\boldsymbol{\Gamma}$ is implementable with complexity 1.



(a) Case 1 (b) Case 2

Figure 5.3: Demonstrating cases 1 and 2 of prof of Lemma 5.3.12. The bidder valuations are demonstrated by blue circles on the real line and the revenue of the two auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are demonstrated by red solid vertical line.

Take any two different auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. We show that $\boldsymbol{\Gamma}^{\mathrm{SL}}_{\boldsymbol{\theta}} \neq \boldsymbol{\Gamma}^{\mathrm{SL}}_{\boldsymbol{\theta}'}$. Let b be the smallest level at which there is $i \in [n]$ such that $\theta_b^i \neq \theta_b'^i$ and among such i choose the largest. There are three cases (see Figure 5.3 for Cases 1 and 2):

1. $i \neq n$: Consider the bid profile $\mathbf{v}^{i,\ell}$ for $\ell = m\theta_b^n$. By the choice of i and the fact that $i \neq n$, we have that $b_n^{\boldsymbol{\theta}}(v_n^{i,\ell}) = b_n^{\boldsymbol{\theta}'}(v_n^{i,\ell}) = b$. On the other hand, $b_i^{\boldsymbol{\theta}}(v_i^{i,\ell}) = s - 1 \geq b$. Therefore, bidder i wins the item in both auctions and pays the b^{th} threshold. So, $\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v}^{i,\ell}) = \theta_b^i \neq \theta_b'^i = \mathrm{Rev}(\boldsymbol{\theta}', \mathbf{v}^{i,\ell})$.

2. $i = n$ and $b \geq 1$: Without loss of generality, assume that $\theta_b^n < \theta_b'^n$. Let $\ell = m\theta_b^n$ and consider $\mathbf{v}^{1,\ell}$. Then $b_n^{\boldsymbol{\theta}}(v_n^{1,\ell}) = b$ and $b_n^{\boldsymbol{\theta}'}(v_n^{1,\ell}) = b'$ for some $b' < b$. So, bidder 1 wins the item in both auctions and pays the threshold that corresponds to the n^{th} bidder's level. Therefore, $\text{Rev}(\boldsymbol{\theta}, \mathbf{v}^{1,\ell}) = \theta_b^1 \neq \theta_{b'}^1 = \text{Rev}(\boldsymbol{\theta}', \mathbf{v}^{1,\ell})$.

3. $i = n$ and $b = 0$: Consider bid profile $\mathbf{e}_n$. In this profile, bidder n wins and pays the absolute reserve price. Therefore, $\text{Rev}(\boldsymbol{\theta}, \mathbf{e}_n) = \theta_0^n \neq \theta_0'^n = \text{Rev}(\boldsymbol{\theta}', \mathbf{e}_n)$.

Therefore, $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}^{\text{SL}} \neq \boldsymbol{\Gamma}_{\boldsymbol{\theta}'}^{\text{SL}}$. Since any element of $\boldsymbol{\Gamma}^{\text{SL}}$ is a multiple of $1/m$, $\boldsymbol{\Gamma}^{\text{SL}}$ is $(m+1, \frac{1}{m})$-admissible. □

Our next theorem is an immediate consequence of Lemma 5.3.12, Theorems 5.2.5 and 5.2.9, and the fact that the revenue of the mechanism at every step is at most 1.

Theorem 5.3.13. *Consider the online auction design problem for the class of s-level auctions with no repeated thresholds, $\mathcal{S}_{s,m}$. Let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\boldsymbol{\Gamma}^{\text{SL}}$ is oracle-efficient with per-round complexity $\text{poly}(n, m, T)$ and has regret*

$$\mathbb{E}\left[\max_{\boldsymbol{\theta} \in \mathcal{S}_{s,m}} \sum_{t=1}^{T} \text{Rev}(\boldsymbol{\theta}, \mathbf{v}_t) - \sum_{t=1}^{T} \text{Rev}(\boldsymbol{\theta}_t, \mathbf{v}_t)\right] \leq O(nm^2\sqrt{T}).$$

5.4 Stochastic Adversaries and Stronger Benchmarks

So far our results apply to general adversaries, where the sequence of adversary actions are arbitrary and where we showed that the payoff of the learner is close to the payoff of the best action in hindsight. Can we make stronger statements about the average payoff of a no-regret learning algorithm when we impose distributional assumptions on the sequence of the adversary?

We start with the easier setting where the actions of the adversary are drawn i.i.d. across all rounds and then we analyze the slightly more complex setting where the actions of the adversary follow a fast-mixing Markov chain. For both settings we show that the average payoff of the learning algorithm is close to the optimal expected payoff, in expectation over the i.i.d. distribution across all rounds in the i.i.d. setting and over the stationary distribution in the Markovian setting.

When applied to the online optimal auction setting, combining these results with approximate optimality results of simple auctions such as s-level auctions or VCG with bidder-specific reserves, we get that the average revenue of our online learning algorithms competes with the revenue achieved by the unrestricted optimal auction for these distributional settings and not only with the best auction within the class over which our algorithms were learning.

5.4.1 Stochastic Adversaries

I.I.D. Adversary One extreme case is to assume that the adversary's action y_t at each iteration is drawn independently and identically from the same unknown distribution F. This leads to

the i.i.d. learning setting. An easy application of the Chernoff-Hoeffding bound yields that for such a learning setting, the average payoff of a no-regret learner converges to the best payoff one could achieve in expectation over the distribution F:

Lemma 5.4.1. *Suppose that $y_1, \ldots, y_T$ are i.i.d. draws from a distribution F. Then for any no-regret learning algorithm, with probability at least $1 - \delta$,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t}[f(x_t, y_t)] \geq \sup_{x \in \mathcal{X}} \mathop{\mathbb{E}}_{y \sim F}[f(x, y)] - \sqrt{\frac{\log(2/\delta)}{2T}} - \frac{\textsc{Regret}}{T}. \tag{5.5}$$

Markovian Adversary Suppose that the choice of the adversary y_t follows a stationary and reversible Markov process based on some transition matrix $P(y, y')$ with a stationary distribution F. Moreover, consider the case where the set $\mathcal{Y}$ is finite. For any Markov chain, the spectral gap γ is defined as the difference between the first and the second largest eigenvalue of the transition matrix P (the first eigenvalue always being 1). We will assume that this gap is bounded away from zero. The spectral gap of a Markov chain is strongly related to its mixing time. In this work we will specifically use the following result of Paulin [220], which is a Bernstein concentration inequality for sums of dependent random variables that are the outcome of a stationary Markov chain with spectral gap bounded away from zero. A Markov chain $y_1, \ldots, y_T$ is stationary if $y_1 \sim F$ where F is the stationary distribution, and is reversible if for any y, y', $F(y)P(y, y') = F(y')P(y', y)$. For simplicity, we focus on stationary chains, though similar results hold for non-stationary chains (see Paulin [220] and references therein).

Theorem 5.4.2 (Paulin [220], Theorem 3.8)**.** *Let $X_1, \ldots, X_z$ be a stationary and reversible Markov chain on a state space Ω, with stationary distribution F and spectral gap γ. Let $g : \Omega \to [0, 1]$, then*

$$\Pr\left[\left|\frac{1}{z}\sum_{i=1}^{z} g(X_i) - \mathop{\mathbb{E}}_{X \sim F}[g(X)]\right| > \epsilon\right] \leq 2\exp\left(-\frac{z\gamma\epsilon^2}{4 + 10\epsilon}\right).$$

Applying this result, we obtain the following lemma (see Appendix A.5 for the proof):

Lemma 5.4.3. *Suppose that the adversary's actions $y_1, \ldots, y_T$ form a stationary and reversible Markov chain with stationary distribution F and spectral gap γ. Then for any no-regret learning algorithm, with probability at least $1 - \delta$:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t}[f(x_t, y_t)] \geq \sup_{x \in \mathcal{X}} \mathop{\mathbb{E}}_{y \sim F}[f(x, y)] - \sqrt{\frac{14\log(2/\delta)}{\gamma T}} - \frac{\textsc{Regret}}{T}. \tag{5.6}$$

Example 5.4.4 (Sticky Markov Chain)**.** *Consider a Markov chain where at every iteration y_t is equal to y_{t-1} with some probability $\rho \geq 1/2$ and with the remaining probability $(1 - \rho)$ it is drawn independently from some fixed distribution F. It is clear that the stationary distribution of this chain is equal to F. We can bound the spectral gap of this Markov chain by the Cheeger*

bound [74]. The Cheeger constant for a finite state, reversible Markov chain is defined and in this case bounded as

$$\Phi = \min_{Q\subseteq\Omega:F(Q)\leq 1/2} \frac{\sum_{y\in Q}\sum_{y'\in Q^c} F(y)P(y,y')}{F(Q)} = \min_{Q\subseteq\Omega:F(Q)\leq 1/2} \frac{\sum_{y\in Q}\sum_{y'\in Q^c} F(y)(1-\rho)F(y')}{F(Q)}$$
$$= \min_{Q\subseteq\Omega:F(Q)\leq 1/2} (1-\rho)\frac{F(Q)\cdot F(Q^c)}{F(Q)} = \min_{Q\subseteq\Omega:F(Q)\leq 1/2} (1-\rho)F(Q^c) \geq \frac{1-\rho}{2}$$

Moreover, by the Cheeger bound we know that $\gamma \geq \frac{\Phi^2}{2} \geq \frac{(1-\rho)^2}{8}$. *Thus we get that for such a sequence of adversary actions, with probability* $1-\delta$,

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t}[f(x_t,y_t)] \geq \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{y\sim F}[f(x,y)] - \frac{4}{1-\rho}\sqrt{\frac{7\log(2/\delta)}{T}} - \frac{\textsc{Regret}}{T} \tag{5.7}$$

5.4.2 Implications for Online Optimal Auction Design

Consider the online optimal auction design problem for a single item and n bidders. Suppose that the adversary who picks the valuation vectors $\mathbf{v}_1,\ldots,\mathbf{v}_T$, is Markovian and that the stationary distribution F of the chain is independent across players, i.e., the stationary distribution is a product distribution $F = F_1 \times \ldots \times F_n$.

Then we know that the optimal auction for this setting is what is known as Myerson's auction [212], which translates the players' values based on some monotone function ϕ, known as the ironed virtual value function and then allocates the item to the bidder with the highest virtual value, charging payments so that the mechanism is dominant-strategy truthful.

A recent result of Morgenstern and Roughgarden [210] shows that level auctions approximate Myerson's auction in terms of revenue. In particular, if distributions F_i are bounded in $[1,H]$, then the class of s-level auctions with $s = \Omega(\frac{1}{\epsilon} + \log_{1+\epsilon} H)$, where the thresholds can be any real numbers, achieves expected revenue at least $(1-\epsilon)$ of the expected optimal revenue of Myerson's auction. Analogously to these results, we prove and use an additive approximation result using thresholds that are in the discretized set $\mathcal{S}_{s,m}$ (defined in Section 5.3.3), rather than the multiplicative guarantees of Morgenstern and Roughgarden [210] that can use any real numbers as thresholds. We use a discretization level $m = \Theta\left(\frac{1}{\epsilon^2}\right)$ and number of levels $s = \Theta\left(\frac{1}{\epsilon}\right)$, to prove that

$$\max_{\boldsymbol{\theta}\in\mathcal{S}_{s,m}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v})] \geq \textsc{Opt}(F) - O(\epsilon), \tag{5.8}$$

where $\textsc{Opt}(F)$ is the optimal revenue achievable by any dominant-strategy truthful mechanism for valuation vector distribution F. At a high level, we first show that one can discretize the support of F to $O(1/\epsilon)$ levels while losing only $O(\epsilon)$ in revenue. We then show that a variant of the class of auctions $\mathcal{S}_{s,1/\epsilon}$, called $\mathcal{R}_{s,1/\epsilon}$ that allows two consecutive thresholds to be equal, approximates the optimal revenue on the discretized valuations. Finally, by using a finer grid, we show that the optimal auction in $\mathcal{S}_{s,m}$ approximates the optimal auction in $\mathcal{R}_{s,1/\epsilon}$, which in turn, approximates the optimal revenue on discretized valuations.

For now, let us defer the proof of this equation to the end of this section. Combining the results in this section with the aforementioned results we get the following theorem:

Theorem 5.4.5 (Competing with Overall Optimal)**.** *Consider the online auction design problem for a single item among n bidders, where the sequence of valuation vectors $\mathbf{v}_1, \ldots, \mathbf{v}_T$ is Markovian, following a stationary and reversible Markov process, with a stationary distribution $F = F_1 \times \ldots \times F_n$ that is a product distribution across bidders, F_is are continuous, and with a spectral gap of $\gamma > 0$. Then the oracle-efficient online learning algorithm (studied in Theorem 5.3.13) which optimizes over the set of s-level auctions $\mathcal{S}_{s,m}$ with $s = \Theta(n^{-1/5}T^{1/10})$ and with a discretization of the threshold levels of size $m = \Theta(n^{-2/5}T^{1/5})$, guarantees the following bound with probability at least $1 - \delta$:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\boldsymbol{\theta}_t}[\mathrm{Rev}(\boldsymbol{\theta}_t, \mathbf{v}_t)] \geq \textsc{Opt}(F) - O\left(\frac{1}{\sqrt{m}}\right) - \sqrt{\frac{14\log(2/\delta)}{\gamma T}} - O\left(\frac{nm^2}{\sqrt{T}}\right)$$
$$\geq \textsc{Opt}(F) - O\left(\frac{n^{1/5}}{T^{1/10}}\right).$$

Example 5.4.6 (Valuation Shocks)**.** *Consider the setting where valuations of players in the beginning of time are drawn from some product distribution $F = F_1 \times \ldots \times F_n$. Then at every iteration with some probability ρ the valuations of all players remain the same as in the previous iteration, while with some probability $1 - \rho$, there is a shock in the market and the valuations of the players are re-drawn from distribution F. As we analyzed in the previous section, the spectral gap of the Markov chain defined by this setting is at least $\frac{(1-\rho)^2}{8}$. Thus we get a regret bound which depends inversely proportionally with the quantity $1 - \rho$.*

Hence, our online learning algorithm achieves revenue that is close to the optimal revenue achievable by any dominant-strategy truthful mechanism for the distribution F. More importantly, it achieves this guarantee even if the valuations of the players are not drawn i.i.d. at every iteration and even if the learner does not know what the distribution F is, when the valuations of the players are going to be re-drawn, or what the rate ρ of shocks in the markets is.

Proof of Equation (5.8)

We set out to prove

$$\max_{\boldsymbol{\theta} \in \mathcal{S}_{s,m}} \mathop{\mathbb{E}}_{\mathbf{v} \sim F}[\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})] \geq \textsc{Opt}(F) - 3\epsilon, \tag{5.9}$$

for $m = 1/\epsilon^2$ and $s = 1/\epsilon$.

Let us first briefly mention two existing approaches to proving similar results using different discretization sets.

Let $\mathcal{R}_{s,m}$ be the set of s-level auctions with discretization level m, as described in Definition 5.3.11, with the exception that consecutive thresholds can be equal. Morgenstern and Roughgarden [210] aim for a multiplicative approximation guarantee and show that when distributions F_i are bounded in $[1, H]$ and $s = \Omega(\frac{1}{\epsilon} + \log_{1+\epsilon} H)$,

$$\max_{\boldsymbol{\theta} \in \mathcal{R}_{s,\infty}} \mathop{\mathbb{E}}_{\mathbf{v} \sim F}[\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})] \geq (1 - \epsilon)\textsc{Opt}(F).$$

Note that this result uses the set of auctions $\mathcal{R}_{s,\infty}$, i.e., the thresholds can be set at any real values and consecutive thresholds may be equal, rather than in the discretization grid $\mathcal{S}_{s,m}$ for a finite m

and no repeated thresholds. On the other hand, Devanur et al. [103] show that the valuations of the bidders can be discretized with only a small loss in the revenue of the optimal auction. That is, if F' is a product distribution obtained by rounding down each $\mathbf{v} \sim F$ to the closest power of $(1-\epsilon)$, then $\textsc{Opt}(F') \geq (1-\epsilon)\textsc{Opt}(F)$.

We use a combination of these results to prove our claim. In particular, we use a variant of the approach of Devanur et al. [103] to obtain an ϵ additive approximation guarantee by discretizing the *valuations of the bidders*, not the auctions, at a discrete grid. We then use a variant of the approach of Morgenstern and Roughgarden [210] to approximate the optimal auction on these valuations using the grid $\mathcal{R}_{s,\infty}$. Since the valuations of the bidders are discretized themselves, we show how to discretize the thresholds of the optimal auction to the set of auctions in $\mathcal{R}_{s,1/\epsilon}$, while losing only a small additive term. Finally, we show how to use a finer grid and approximate the revenue of the optimal auction using level auctions that have distinct thresholds, i.e., $\mathcal{S}_{s,1/\epsilon^2}$.

First, let F' be a product distribution obtained by rounding down each $\mathbf{v} \sim F$ to the closest multiple of ϵ. We show that $\textsc{Opt}(F') \geq \textsc{Opt}(F) - \epsilon$. The proof is an analogue of Lemma 5 of [103] for additive approximation.

Lemma 5.4.7. *Given any product distribution $F = F_1 \times \ldots \times F_n$, let F' be the distribution obtained by rounding down the values from F to the closest multiple of ϵ. Then $OPT(F') \geq OPT(F) - \epsilon$.*

Proof. Let M be the optimal Myerson auction for F and let M' be the following mechanism for allocating items to $\mathbf{v}' \sim F'$: Take $\mathbf{v}$ such that $1 - F_i(v_i) = 1 - F'_i(v'_i)$ for all $i \in [n]$ and allocate the item according to the outcome of mechanism M on $\mathbf{v}$. Charge the winner the minimum value above which it would remain a winner.

To analyze the revenue of M', we will think of the expected revenue in terms of quantiles. The quantile of a distribution F_i is defined as $c_i(v_i) = 1 - F_i(v_i)$. So instead of thinking of an auction in terms of the allocation as a function of values, we can think of it in terms of quantiles and we can write $v_i(c) = c_i^{-1}(c)$ as the value that corresponds to quantile c. Then we can express the expected revenue as $\mathbb{E}_{\mathbf{c}}[\mathrm{Rev}(M, \mathbf{v}(\mathbf{c})]$, where each quantile c_i is drawn uniformly in $[0,1]$.

Consider any vector of quantiles $\mathbf{c} \in [0,1]^n$ and let $\mathbf{v}$ and $\mathbf{v}'$ be the values in F and F' corresponding to these quantiles, i.e., $1 - F_i(v_i) = 1 - F'_i(v'_i) = c_i$ for all $i \in [n]$. Note that, allocation of M on $\mathbf{v}$ is the same as the allocation of M' on $\mathbf{v}'$. Moreover, the payment of a truthful mechanism is the threshold value of the winner above which he still remains allocated. Therefore, for any such quantile the payment in F and F' differ by at most ϵ. This proves that $\textsc{Opt}(F') \geq \textsc{Opt}(F) - \epsilon$. □

Next, we show that there is $\boldsymbol{\theta} \in \mathcal{R}_{1/\epsilon,\infty}$ such that $\mathbb{E}_{\mathbf{v}\sim F'}[\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})] \geq \textsc{Opt}(F') - \epsilon$. This is an analogue of Theorem 3.4 in Morgenstern and Roughgarden [210], but for additive approximation, which enables us to drop some strong assumptions made in Morgenstern and Roughgarden [210] on the support of the distribution.

Lemma 5.4.8. *Consider any product distribution $F = F_1 \times \ldots \times F_n$, where each F_i has support in $[0,1]$. We have that:*

$$\max_{\boldsymbol{\theta} \in \mathcal{R}_{1/\epsilon,\infty}} \mathbb{E}_{\mathbf{v}\sim F}[\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})] \geq \textsc{Opt}(F) - \epsilon. \tag{5.10}$$

Proof. For each i, let $\phi_i(\cdot)$ be the ironed virtual valuation function for bidder i with respect to F_i (see [150]). These $\phi_i(\cdot)$ are non-decreasing functions. Let $\boldsymbol{\theta}$ be such that $\theta_b^i = \phi_i^{-1}(b \cdot \epsilon)$ for $b \in \{0, 1, \ldots, s-1\}$ (where the inverse is defined as the left-most value v for which $\phi_i(v) = b \cdot \epsilon$). Observe that with such a vector of thresholds, the s-level auction is essentially approximating the virtual value functions to within an epsilon error, and then allocating to the player with the highest approximate virtual value function. In particular, observe that a player i in this s-level auction is assigned level b if his ironed virtual value $\phi_i(v_i)$, is in interval $[b\epsilon, (b+1)\epsilon)$, or in $[b\epsilon, 1]$ if $b = s - 1$.

Next, we show that for any $\mathbf{v}$, $\text{Rev}(\boldsymbol{\theta}, \mathbf{v}) \geq \textsc{Opt}(F) - \epsilon$. Consider $\mathbf{v} \sim F$ and let i^* and i' be the winners in $\boldsymbol{\theta}$ and the Myerson optimal auctions respectively. Moreover, observe that in both auctions the allocation of a player can be determined as a function of his ironed virtual value (rather than directly his value). Thus we can conclude by Theorem 3.18 of [150], that the expected revenue of both auctions is equal to their expected ironed virtual value (not only upper bounded by it). So, we have that:

$$\mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\text{Rev}(\boldsymbol{\theta}, \mathbf{v})] = \mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\phi_{i^*}(v_{i^*})] \quad \text{and} \quad \textsc{Opt}(F) = \mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\phi_{i'}(v_{i'})].$$

Now, consider the winners i^* and i'. Note that if i^* was the unique bidder at the highest bucket under $\boldsymbol{\theta}$ (there were no ties to be broken lexicographically), then i^* also has the highest ironed virtual valuation, so $i^* = i'$. On the other hand, if i' was tied with i^*, then $\phi_{i'}(v_{i'}) - \phi_{i^*}(v_{i^*}) \leq \epsilon$. So, overall we have

$$\mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\text{Rev}(\boldsymbol{\theta}, \mathbf{v})] = \mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\phi_{i^*}(v_{i^*})] \geq \mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\phi_{i'}(v_{i'})] - \epsilon = \textsc{Opt}(F) - \epsilon.$$

□

We can now combine the above two Lemmas to show Equation (5.9). From Lemma 5.4.8, starting from any product distribution F, we can first round down values to the nearest multiple of ϵ to get a distribution F', such $\textsc{Opt}(F') \geq \textsc{Opt}(F) - \epsilon$. Then we can apply Lemma 5.4.8, for distribution F', to show that

$$\max_{\boldsymbol{\theta}\in\mathcal{R}_{1/\epsilon,\infty}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F'}[\text{Rev}(\boldsymbol{\theta}, \mathbf{v})] \geq \textsc{Opt}(F') - \epsilon \geq OPT(F) - 2\epsilon.$$

Next, observe that since the distribution F' is only supported at values that are multiples of ϵ, in the above maximization over threshold levels, it suffices to optimize over thresholds that are multiples of ϵ. For any other threshold, θ_b^i, which is not a multiple of ϵ, we can rounded up to the nearest multiple of ϵ. This does not change the allocation of any realized value and it can only increases the payment. Thus:

$$\max_{\boldsymbol{\theta}\in\mathcal{R}_{1/\epsilon,1/\epsilon}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F'}[\text{Rev}(\boldsymbol{\theta}, \mathbf{v})] = \max_{\boldsymbol{\theta}\in\mathcal{R}_{1/\epsilon,\infty}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F'}[\text{Rev}(\boldsymbol{\theta}', \mathbf{v})] \geq \textsc{Opt}(F) - 2\epsilon.$$

Next, we show that without loss of generality, we can consider those auctions in $\boldsymbol{\theta} \in \mathcal{R}_{1/\epsilon,1/\epsilon}$ where for each bidder i, only θ_0^i can be set at value 0 and no other threshold. Consider $\boldsymbol{\theta}$ such that $\theta_0^i =, \ldots, \theta_b^i = 0$. Note that no payment in such an auction falls in the bucket $b-1$ of bidder $i' < i$ or bucket b of $i' > i$. Therefore, one can remove thresholds $\theta_1^i, \cdots, \theta_b^i$ and the

corresponding thresholds for bidders i' without changing the allocation or payment on any valuation. Therefore, in the remainder of this proof, we assumes that $\mathcal{R}_{1/\epsilon,1/\epsilon}$ is restricted to auctions with no repeated thresholds on value 0.

Note that the thresholds created above fall in the set $\mathcal{R}_{1/\epsilon,1/\epsilon}$, that is they may have consecutive thresholds that are equal (however, not on value 0). Instead, consider level auctions at a finer grid value, $\mathcal{S}_{1/\epsilon,1/\epsilon^2}$. For any $\boldsymbol{\theta} \in \mathcal{R}_{1/\epsilon,1/\epsilon}$, consider a $\boldsymbol{\theta}' \in \mathcal{S}_{1/\epsilon,1/\epsilon^2}$ such that the equal threshold in $\boldsymbol{\theta}$ are spread in the $\frac{1}{\epsilon}$ discretization levels between the two discretization levels of $\boldsymbol{\theta}$. That is, for any bidder i and any $\ell \in [1/\epsilon]$, all the thresholds of bidder i that are equal to $\ell\epsilon$ are spread to get distinct values $\{(\ell-1)\epsilon+\epsilon^2, (\ell-1)\epsilon+2\epsilon^2, \cdots, \ell\epsilon\}$ in $\boldsymbol{\theta}'$. Since auction $\boldsymbol{\theta}$ has $\frac{1}{\epsilon}$ levels, all levels are distinct in $\boldsymbol{\theta}'$, and since $\boldsymbol{\theta}$ has at most one threshold on value 0, no negative value thresholds are created in $\boldsymbol{\theta}'$. Moreover, because the valuations in F' are themselves discretized at multiples of ϵ, no bid changes level and the allocation in $\boldsymbol{\theta}$ is the same as $\boldsymbol{\theta}'$. So, the revenue of any valuation drops by at most ϵ. Therefore,

$$\max_{\boldsymbol{\theta}\in\mathcal{S}_{1/\epsilon,1/\epsilon^2}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F'}[\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v})] \geq \max_{\boldsymbol{\theta}\in\mathcal{R}_{1/\epsilon,1/\epsilon}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F'}[\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v})] - \epsilon.$$

Finally, since valuations of F are rounded down to valuations in F', we have that under F, any s-level auction can only yield higher revenue than under F', because F has point-wise higher values than F. Hence:

$$\max_{\boldsymbol{\theta}\in\mathcal{S}_{1/\epsilon,1/\epsilon^2}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F}[\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v})] \geq \max_{\boldsymbol{\theta}\in\mathcal{S}_{1/\epsilon,1/\epsilon^2}} \mathop{\mathbb{E}}_{\mathbf{v}\sim F'}[\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}) \geq \textsc{Opt}(F) - 3\epsilon.$$

5.5 Approximate Oracles and Approximate Regret

As we saw, the Oracle-Based Generalized FTPL algorithm requires an oracle to choose an action with a total payoff that is within a small additive error of that of the optimal action. In this section, we extend our main results regarding the Oracle-Based Generalized FTPL algorithm to work with oracles that return an action whose payoff is only a constant approximation of that of the optimal actions. An offline approximation oracle is defined formally as follows:

Definition 5.5.1 (Offline Approximation Oracle)**.** *An* offline approximation oracle *for a set of learner's actions $\mathcal{X}$ and function f, C-$\textsc{Opt}_{(f,\mathcal{X})}$ where $C \leq 1$, is any algorithm that receives as input a weighted set of adversary actions $S = \{(w_\ell, y_\ell)\}_{\ell\in\mathcal{L}}$, $w_k \in \mathbb{R}^+$, $y_k \in \mathcal{Y}$, and returns $x \in \mathcal{X}$, such that*

$$\sum_{(w,y)\in S} wf(x,y) \geq C \max_{x\in\mathcal{X}} \sum_{(w,y)\in S} wf(x,y).$$

Similarly below we will use notation $\textsc{Opt}_{(f,\mathcal{X})}$ to represent an offline oracle $\textsc{Opt}$ defined in Section 5.2.2 for function f and set $\mathcal{X}$, making the dependence explicit.

As discussed earlier, access to an oracle $\textsc{Opt}$ that has a small additive approximation error is needed for achieving no-regret results. That is, using standard online-to-batch reductions [71, 94], one can turn a polynomial time online no-regret algorithm into a polynomial time additive approximation scheme for the offline problem. So, when the best approximation for a problem is obtained through a C-approximation oracle for general C, there is no hope for achieving

no-regret results. Instead Kakade et al. [168] introduced an alternative measure of regret, called C-REGRET, for competing with an offline approximation algorithm. Formally, the C-REGRET of an online maximization problem is defined as

$$C\text{-REGRET} = \mathbb{E}\left[C \max_{a \in \mathcal{A}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right].$$

Note that for FPTAS algorithm, where $C = 1 - \epsilon$ for any desirable ϵ, using $\epsilon = 1/\text{poly}(T)$ recovers a no-regret guarantee. Below we consider several types of constant approximation oracles, such as approximation through relaxation of the objective and approximation oracles that stem from Maximal-in-Range (MIR) algorithms, and show that using the Oracle-Based Generalized FTPL algorithm with the approximation oracles obtains vanishing C-REGRET.

5.5.1 Approximation through Relaxation

A large class of approximation algorithms achieve their approximation guarantees by exactly optimizing a relaxation of the objective functions. More formally, if there is a function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, such that $Cf(x, y) \leq F(x, y) \leq f(x, y)$ and there is an offline oracle $\text{OPT}_{(F,\mathcal{X})}$ (with $\epsilon = 0$ for simplicity), then it is clear that any online algorithm for F is also online algorithm with vanishing C-REGRET for f. This result is more formally stated below.

Theorem 5.5.2. *Let F be a functions such that for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, $f(x, y) \geq F(x, y) \geq Cf(x, y)$. Let $\mathbf{\Gamma}^F \in [0, 1]^{|\mathcal{X}| \times N}$ be (κ, δ)-admissible and implementable for function F with complexity $g(N, T)$. Let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \sqrt{\delta/(2T\kappa)}$. Then, the Oracle-Based Generalized FTPL algorithm with D, datasets that implement $\mathbf{\Gamma}^F$, and $\text{OPT}_{(F,\mathcal{X})}$ is oracle-efficient with per-round complexity $O(T + g(N, T))$ and regret*

$$C\text{-REGRET} \leq 2N\sqrt{2T\kappa/\delta}.$$

A similar observation was made by Balcan and Blum [31] regarding approximation algorithms that use linear optimization as a relaxation and therefore can be efficiently optimized by the standard FTPL algorithm of Kalai and Vempala [169]. Our work extends this observation to any relaxation of function f that has an FPTAS and an admissible and implementable translation matrix.

Roughgarden and Wang [239] as a Relaxation. The approach of Roughgarden and Wang [239] for achieving a $1/2$-regret for single-item second price auctions with bidder-specific reserves, falls exactly in the relaxation approximation framework. They give a relaxed objective which admits a polynomial time offline oracle and which is always within a factor 2 from the original objective. Then they run an oracle based online learning algorithm for the relaxed objective. However, in their case the relaxed objective corresponds to an online linear optimization problem and can be solved with the standard FTPL algorithm of Kalai and Vempala [169]. The theorem above shows that the same approach extends even if the relaxed objective does not reduce to an online linear optimization problem but to a problem that can be tackled by our Generalized FTPL, providing a potential avenue for obtaining vanishing C-REGRET for values of $C \geq \frac{1}{2}$.

5.5.2 Approximation by Maximal-in-Range Algorithms

Another interesting class of approximation algorithms is Maximal-in-Range (MIR) algorithms. An MIR algorithm commits to a set of feasible solutions $\mathcal{X}' \subseteq \mathcal{X}$ independently of the input to the algorithm and outputs the best solution $x \in \mathcal{X}'$. That is, an MIR C-approximation algorithm forms an approximation oracle $C\text{-}\textsc{Opt}_{(f,\mathcal{X})}(S) = \textsc{Opt}_{(f,\mathcal{X}')}(S, 0)$ for any S. Consider an MIR approximation algorithm and $\mathbf{\Gamma}^{\mathcal{X}}$ translation matrix that is admissible and implementable for $\mathcal{X}$. Clearly, $\mathbf{\Gamma}^{\mathcal{X}}$ restricted to the set of rows in $\mathcal{X}'$ (denoted by $\mathbf{\Gamma}^{\mathcal{X}'}$) is also admissible and implementable for any $\mathcal{X}' \subseteq \mathcal{X}$. In fact $\mathbf{\Gamma}^{\mathcal{X}'}$ is (κ', δ') admissible for $\kappa' \leq \kappa$ and $\frac{1}{\delta'} \leq \frac{1}{\delta}$. Thus even better regret guarantees could be achievable if one uses the smaller admissibility quantities of matrix $\mathbf{\Gamma}^{\mathcal{X}'}$. Therefore, an MIR C-approximation algorithm leads to an efficient online algorithm with vanishing C-$\textsc{Regret}$. In Section 5.6.1, we demonstrate an example where a better approximate regret bound is obtained using smaller admissibility quantities.

More formally we have:

Theorem 5.5.3. *Let $\mathbf{\Gamma}^{\mathcal{X}'} \in [0,1]^{|\mathcal{X}'| \times N}$ be (κ', δ')-admissible and implementable with complexity $g(N,T)$. Let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \sqrt{\delta'/(2T\kappa')}$. Then, the Oracle-Based Generalized FTPL algorithm with D, datasets that implement $\mathbf{\Gamma}^{\mathcal{X}'}$ (or equivalently datasets that implement $\mathbf{\Gamma}^{\mathcal{X}}$) and an MIR approximation oracle $\textsc{Opt}_{(f,\mathcal{X}')}$ is oracle-efficient with per-round complexity $O(T + g(N,T))$ and regret*

$$C\text{-}\textsc{Regret} \leq 2N\sqrt{2T\kappa'/\delta'}.$$

5.6 Additional Applications and Connections

In this section, we discuss an application of our oracle efficient learning approach to the problem of online welfare maximization in multi-unit auctions, to no-regret learning in simultaneous second price auctions, and discuss the connections between our notions of admissibility and implementability with other statistical measures of complexity from learning theory.

For readability, we use superscript (t), as opposed to the subscript t, to index the time step throughout this section.

5.6.1 Fully Efficient Online Welfare Maximization in Multi-Unit Auctions

In this section, we consider the class of single-item multi-unit auctions that has $1/2$-approximation Maximal-in-Range (MIR) algorithm. We show how the results of Section 5.5 can be applied to this problem to achieve a *truly polynomial time* online algorithm with vanishing $\frac{1}{2}$-$\textsc{Regret}$.

We consider an online learning variant of an n-bidder *multi-unit* environment, better modeled as a set of s identical items. Each bidder i has a monotonic valuation function $v_i : [s] \to [0,1]$. In other words, bidder i has *marginal valuation* $\mu_i(\ell)$ for receiving its ℓ^{th} item and the total utility of bidder i for receiving q_i items is $v_i(q_i) = \sum_{\ell=1}^{q_i} \mu_i(\ell)$. Here we consider the case where s is much larger than n and the goal is to find an allocation $\mathbf{q} \in \mathbb{Z}^n$, subject to $\sum_{i=1}^n q_i = s$, that maximizes the total welfare $\sum_{i=1}^n v_i(q_i)$ in time polynomial in n and $\log(s)$. In the online learning setting, every day t a fresh set of bidders arrive with new valuations v_i^t and the learner

commits to an allocation of the units of the item to the players, prior to seeing the valuations. The goal of the learner is to pick an allocation each day that competes with the best in hindsight allocation.

It is not hard to see that the offline welfare maximization problem in this setting corresponds to the Knapsack problem, where each player has a valuation equal to the average value in hindsight, i.e., $\frac{1}{T}\sum_{t=1}^{T} v_i^{(t)}(\cdot)$. So, dynamic programming can be used to compute a welfare-maximizing allocation in time polynomial in n and s. Dobzinski and Nisan [110] introduced a $1/2$-approximation MIR algorithm for this problem. At a high level, the algorithm proceeds by dividing the set of items to n^2 bundles of size s/n^2.[8] Then, the MIR algorithm chooses the best allocation from the set of allocations (range of the algorithm) where all the items in one bundle are allocated to the same bidders. This algorithm is effectively solving a knapsack problem over n^2 items and can be implemented in time polynomial in n and $\log(s)$.

We show how to construct a matrix $\mathbf{\Gamma}^{\text{MU}}$ that is admissible and implementable for unrestricted n-bidder s-unit auctions. We then use Theorem 5.5.3 to obtain an online algorithm with vanishing $\frac{1}{2}$-REGRET that runs in time $\text{poly}(n, T, \log s)$.

Construction of $\mathbf{\Gamma}$: Let $\mathbf{\Gamma}^{\text{MU}}$ be a matrix with n columns, such that for any allocation $\mathbf{q}$ and any column j, $\Gamma^{\text{MU}}_{\mathbf{q}j} = q_j/s$. Let $\mathbf{\Gamma}^{\text{MU}'}$ be the restriction of matrix $\mathbf{\Gamma}^{\text{MU}}$ to the rows that represent the range of the $1/2$-approximation algorithm of Dobzinski and Nisan [110].

Clearly, for any $\mathbf{q} \neq \mathbf{q}'$ we have $\mathbf{\Gamma}^{\text{MU}}_{\mathbf{q}} \neq \mathbf{\Gamma}^{\text{MU}}_{\mathbf{q}'}$. So, $\mathbf{\Gamma}^{\text{MU}}$ is $(s+1, \frac{1}{s})$-admissible. Moreover, observe that the matrix $\mathbf{\Gamma}^{\text{MU}'}$ restricted to the range of the $1/2$ approximation algorithm of Dobzinski and Nisan [110] has much better admissibility constants. In particular, the number of different entries within each column is at most $\kappa' = n^2 + 1$, since each player receives allocations that are multiples of s/n^2 and there are at most $n^2 + 1$ such multiples. Moreover, the minimum non-zero difference of the column entries, between any two such bundled allocations, is at least $\delta' \geq \frac{s}{n^2}\frac{1}{s} = \frac{1}{n^2}$. Therefore matrix $\mathbf{\Gamma}^{\text{MU}'}$ is $\left(n^2+1, \frac{1}{n^2}\right)$-admissible.

It is not hard to see that $\mathbf{\Gamma}^{\text{MU}}$ is also implementable with complexity 1. For any column j, consider the bid profile $\mathbf{v}^j$ where bidder j's marginal valuation for any item is $1/s$ and all other bidders have 0 valuation for any number of items. That is, $\mu_j(\ell) = 1/s$ and $\mu_i(\ell) = 0$ for all ℓ and $i \neq j$. The welfare of any allocation on this bid profile is the utility of bidder j, which is $q_j/s = \Gamma^{\text{MU}}_{\mathbf{q}j}$. Therefore, $\mathbf{\Gamma}^{\text{MU}}$ is implementable with complexity 1 using datasets $S_j = \{(1, \mathbf{v}^j)\}$ for all $j \in [n]$. As a result, $\mathbf{\Gamma}^{\text{MU}'}$ is also implementable with complexity 1 using the same datasets.

Theorem 5.6.1. *Consider the problem of welfare maximization in s-unit auctions. Let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \sqrt{\delta'/(2T\kappa')} = \sqrt{\frac{1}{2Tn^2(n^2+1)}}$. Then, for any sequence of valuation functions $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(T)}$, the Oracle-Based Generalized FTPL algorithm with datasets that implement matrix $\mathbf{\Gamma}^{\text{MU}'}$, distribution D, and the $1/2$-approximate MIR algorithm of [110] as an oracle, runs in per-round time $\text{poly}(n, \log(s), T)$ and plays the sequence of allocations*

[8] If s is not a multiple of n^2, one can add extra items with marginal value of 0 to all bidders.

$\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(T)}$, *such that*

$$\begin{aligned}\frac{1}{2}\text{-}\textsc{Regret} &= \mathbb{E}\left[\frac{1}{2}\left(\max_{\substack{\mathbf{q}\in\mathbb{Z}^n:\\ \|\mathbf{q}\|_1=s}} \sum_{t=1}^{T}\sum_{i=1}^{n} v_i^{(t)}(q_i)\right) - \sum_{t=1}^{T}\sum_{i=1}^{n} v_i^{(t)}(q_i^{(t)})\right] \\ &\leq n \cdot 2n\sqrt{2T(n^2+1)n^2} \\ &\leq O(n^4\sqrt{T}).\end{aligned}$$

In the above theorem, the extra factor of n in the regret as compared to that implied by Theorem 5.5.3, is due to the fact that the maximum welfare in this problem is upper bounded by n.

5.6.2 Oracle Efficient Online Bidding in Simultaneous Second Price Auctions

In this section, we answer an open problem raised by Daskalakis and Syrgkanis [94] regarding the existence of an oracle-based no-regret algorithm for optimal bidding in Simultaneous Second-Price Auctions. We show that our Oracle-Based Generalized FTPL algorithm used with an appropriate implementable and admissible translation matrix can be used to obtain such an algorithm.

A *Simultaneous Second-Price Auction (SiSPA)* [47, 80, 118] is a mechanism for allocating k items to n bidders. Each bidder $i \leq n$ submits k simultaneous bids denoted by a vector of bids $\mathbf{b}_i$. The mechanism allocates each item using a second-price auction based on the bids solely submitted for this item, while breaking ties in favor of bidders with larger indices. For each item j, the winner is charged p_j, the second highest bid for that item. Each bidder i has a fixed combinatorial valuation function $v_i : \{0,1\}^k \to [0,1]$ over bundles of items. Then, the total utility of bidder i who is allocated the bundle $\mathbf{q}_i \in \{0,1\}^k$ is $v_i(\mathbf{q}_i) - \mathbf{p} \cdot \mathbf{q}_i$, where $\mathbf{p}$ is the vector of second largest bids across all items.

We consider the problem of optimal bidding in a SiSPA from the perspective of the first bidder. Hereafter, we drop the indices of the players from the notation. From this perspective, the utility of the bidder only depends on its bid $\mathbf{b}$ and the threshold vector $\mathbf{p}$ of the second largest bids. The online optimal bidding problem is defined as follows.

Definition 5.6.2 (Online Bidding in SiSPAs [254])**.** *At each time step t, the player picks a bid vector $\mathbf{b}^{(t)}$ and an adversary picks a threshold vector $\mathbf{p}^{(t)}$. The player wins the bundle of items $\mathbf{q}(\mathbf{b}^{(t)}, \mathbf{p}^{(t)})$, with $q_j(\mathbf{b}^{(t)}, \mathbf{p}^{(t)}) = \mathbf{1}_{(b_j^{(t)} > p_j^{(t)})}$ and gets the utility*

$$u(\mathbf{b}^{(t)}, \mathbf{p}^{(t)}) = v\left(\mathbf{q}(\mathbf{b}^{(t)}, \mathbf{p}^{(t)})\right) - \mathbf{p}^{(t)} \cdot \mathbf{q}.$$

We consider this problem under the *no-overbidding condition* that requires that for any bundle $\mathbf{q}$, the sum of bids over items in $\mathbf{q}$ does not exceed the bidder's valuation for $\mathbf{q}$, i.e., $\mathbf{b}^{(t)} \cdot \mathbf{q} \leq v(\mathbf{q})$, for all $\mathbf{q} \in \{0,1\}^k$. Similar no-overbidding assumptions are used in the previous work to prove that no-regret learning in second-price auctions has good welfare guarantees [80, 118].

We consider the online bidding problem where for any $\mathbf{q}$, the valuation $v(\mathbf{q}) \in \{0, 1/m, \ldots, m/m\}$ and for any item $j \leq k$, b_j is a multiple of $1/m$. We represent by $\mathcal{B}_m$ the class of all such bid vectors that satisfy the no-overbidding condition for $v(\cdot)$. Note that the assumption on the valuation function is not restrictive. That is, for any valuation function $v(\cdot)$, one can round down its values to the closest multiple of $1/m$, while losing at most $1/m$ utility. Moreover, a similar discretization for the bid vectors was used by Daskalakis and Syrgkanis [254] for studying offline and online optimal bidding in SiSPAs.

Next, we show how to construct an implementable and admissible translation matrix for $\mathcal{B}_m$.

Construction of $\mathbf{\Gamma}$: Let $\mathbf{\Gamma}^{\text{OB}}$ be a matrix with k columns and $|\mathcal{B}_m|$ rows that are equal to bid vectors, i.e., $\mathbf{\Gamma}^{\text{OB}}_{\mathbf{b}} = \mathbf{b}$.

The next lemma shows that $\mathbf{\Gamma}^{\text{OB}}$ is admissible and implementable.

Lemma 5.6.3. *$\mathbf{\Gamma}^{\text{OB}}$ is $(m+1, 1/m)$-admissible and implementable with complexity m.*

Proof. Since $\mathbf{\Gamma}^{\text{OB}}_{\mathbf{b}} = \mathbf{b}$ and $\mathbf{b}$ is descretized, $\mathbf{\Gamma}^{\text{OB}}$ is $(m+1, 1/m)$-admissible. Next, we argue that $\mathbf{\Gamma}^{\text{OB}}$ is also implementable. Consider column j of $\mathbf{\Gamma}^{\text{OB}}$. We show that datasets S_j for all j that is only supported on threshold vectors where all but the jth thresholds are set to 1, implement $\mathbf{\Gamma}^{\text{OB}}$. Specifically, for $\ell = 0, 1, \ldots, m-1$, let $\mathbf{p}^j_\ell = (\ell/m)\mathbf{e}_j + \sum_{j' \neq j} \mathbf{e}_{j'}$. Note that the utility of playing bid $\mathbf{b}$ against $\mathbf{p}^j_\ell$ is $u(\mathbf{b}, \mathbf{p}^j_\ell) = \big(v(\mathbf{e}_j) - \ell/m\big)\mathbf{1}_{(b_j > \ell/m)}$. We set the weight corresponding to $\mathbf{p}^j_\ell$ to

$$w^j_\ell = \begin{cases} \frac{1}{m} \cdot \frac{1}{v(\mathbf{e}_j) - \ell/m} & \text{if } \ell/m < v(\mathbf{e}_j) \\ 0 & \text{otherwise.} \end{cases}$$

Since $b_j \leq v(\mathbf{e}_j)$ for any $\mathbf{b}$, we have

$$\begin{aligned} \sum_{\ell=0}^{m-1} w^j_\ell \, u(\mathbf{b}, \mathbf{p}^j_\ell) &= \sum_{\ell=0}^{m-1} \frac{1}{m} \cdot \frac{1}{v(\mathbf{e}_j) - \ell/m} \cdot \Big(v(\mathbf{e}_j) - \ell/m\Big)\mathbf{1}_{(b_j > \ell/m)} \\ &= \sum_{\ell=0}^{m-1} \frac{1}{m} \mathbf{1}_{(b_j > \ell/m)} = \sum_{\ell=0}^{mb_j - 1} \frac{1}{m} \\ &= b_j = \mathbf{\Gamma}^{\text{OB}}_{\mathbf{b}j}. \end{aligned}$$

Thus, indeed $S_j = \{(w^j_\ell, \mathbf{p}^j_\ell)\}_\ell$ implements $\mathbf{\Gamma}^{\text{OB}}$. Note that $|S_j| \leq m$, so $\mathbf{\Gamma}^{\text{OB}}$ is implementable with complexity m. □

The next theorem is a direct consequence of Lemma 5.6.3 and Theorems 5.2.5 and 5.2.9.

Theorem 5.6.4. *Consider the problem of Online Bidding in SiSPAs. Let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\text{OB}}$ is oracle-efficient with per-round complexity $\text{poly}(k, m, T)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{b} \in \mathcal{B}_m} \sum_{t=1}^{T} u(\mathbf{b}, \mathbf{p}^{(t)}) - \sum_{t=1}^{T} u(\mathbf{b}^{(t)}, \mathbf{p}^{(t)})\right] \leq O(km\sqrt{T}).$$

5.6.3 Universal Identification Sequences

There is an interesting connection between our definitions of admissibility and implementability and a statistical measure of complexity from learning theory, called the *Universal Identification Sequences*.

Definition 5.6.5 (Universal Identification Sequences [131])**.** *Consider a domain $\mathcal{Z}$ and a class of functions $\mathcal{F}$ mapping from $\mathcal{Z}$ to $\{0,1\}$. A set of unlabeled instances $\mathcal{Z}' \subseteq \mathcal{Z}$ is said to* distinguish *function $f \in \mathcal{F}$ if f is the only function that is consistent with the labeling on $\mathcal{Z}'$ produced by f. A set of unlabeled instances $\mathcal{Z}'$ is called a* universal identification sequence *if it distinguishes every $f \in \mathcal{F}$.*

Any universal identification sequence of $\mathcal{F}$ can be used to construct a translation matrix that is admissible and implementable. Consider a matrix $\mathbf{\Gamma}^{\mathcal{F}}$, whose rows are indexed by $\mathcal{F}$ and columns are indexed by $\mathcal{Z}'$, such that $\mathbf{\Gamma}^{\mathcal{F}}_{fz} = f(z)$ for any $f \in \mathcal{F}$ and $z \in \mathcal{Z}'$. By the definition of universal identification sequence for any two functions, $f, f' \in \mathcal{F}$ there is $z \in \mathcal{Z}'$, such that $f(z) \neq f'(z)$, i.e., $\mathbf{\Gamma}^{\mathcal{F}}_{f} \neq \mathbf{\Gamma}^{\mathcal{F}}_{f'}$. As a results $\mathbf{\Gamma}^{\mathcal{F}}$ is $(2,1)$-admissible. Moreover, the columns of $\mathbf{\Gamma}^{\mathcal{F}}$ correspond to the value of functions applied to $z \in \mathcal{Z}'$. Therefore, $\mathbf{\Gamma}^{\mathcal{F}}$ is implementable with complexity 1 using datasets $S_z = \{(1, z)\}$ for all $z \in \mathcal{Z}'$. That is, the *length of a universal identification sequence* is an upper bound on the number of columns needed to create a translation matrix that is admissible and implementable for a class of binary functions. Examples of function classes with polynomial-length universal identification sequences include logarithmic-depth read-once majority formulas and logarithmic-depth read-once positive NAND formulas [131]. The next corollary is a direct consequence of Theorems 5.2.5 and 5.2.9.

Corollary 5.6.6. *Consider a domain $\mathcal{Z}$ and a class of binary functions $\mathcal{F}$ with a universal identification sequence $\mathcal{Z}'$ of length d. Let D be the uniform distribution as described in Theorem 5.2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implements $\mathbf{\Gamma}^{\mathcal{F}}$ is oracle-efficient with per-round complexity $\mathrm{poly}(d, T)$ and has regret*

$$\mathbb{E}\left[\max_{f\in\mathcal{F}} \sum_{t=1}^{T} f(z^{(t)}) - \sum_{t=1}^{T} f^{(t)}(z^{(t)})\right] \leq O(d\sqrt{T}).$$

Our condition on the existence of admissible and implementable matrices is very similar to the existence of short universal identification sequences. In particular, when f is a boolean function these two notions are equivalent. However, our Oracle-Based Generalized FTPL algorithm goes beyond the use of binary functions and universal identification sequences. In particular, we applied our results to obtain no-regret algorithms for several real-valued function classes. Furthermore, we introduced implementable translation matrices where each column corresponds to a complex weighted set of adversary's actions, rather than, columns that correspond to individual adversary's actions.

Chapter 6

Online Learning with a Hint

6.1 Introduction

The need for robust learning algorithms has led to the creation of online learning algorithms with performance guarantees that hold even when the environment that the learner performs in changes adversarially. A canonical example of this framework is the online linear optimization problem. In this setting, a player attempts to minimize an online adversarial sequence of linear loss functions while incurring a small *regret*, compared to the best offline solution. Many online algorithms exist that are designed to have a regret of $O(\sqrt{T})$ in the worst case and it has been known that one cannot avoid a regret of $\Omega(\sqrt{T})$ in the worst case. Having been designed to perform well in adversarial environments, however, these algorithms can be overly pessimistic for some day-to-day applications where the changes in the environment may be undesirable but not necessarily adversarial. In particular, the online learning framework does not account for the fact that the player may have side-information that allows him to anticipate the upcoming loss functions and evade the $\Omega(\sqrt{T})$ regret. In this chapter, we go beyond this worst case analysis and consider online linear optimization when additional information in the form of a function that is correlated with the loss is presented to the player.

More formally, online convex optimization [153, 274] is a T-round repeated game between a player and an adversary. On each round, the player chooses an action x_t from a convex set of feasible actions $\mathcal{K} \subseteq \mathbb{R}^d$ and the adversary chooses a convex bounded loss function f_t. Both choices are revealed and the player incurs a loss of $f_t(x_t)$. The player then uses its knowledge of f_t to adjust its strategy for the subsequent rounds. The player's goal is to accumulate a small loss compared to the best fixed action in hindsight. This value is called *regret* and is a measure of success of the player's algorithm.

When the adversary is restricted to Lipschitz loss functions, several algorithms are known to guarantee $O(\sqrt{T})$ regret [153, 169, 274]. If we further restrict the adversary to strongly convex loss functions, the regret bound improves to $O(\log(T))$ [158]. However, when the loss functions are linear, no online algorithm can have a regret of $o(\sqrt{T})$ [70]. In this sense, linear loss functions are the most difficult convex loss functions to handle [274].

In this chapter, we focus on the case where the adversary is restricted to linear Lipschitz loss functions. More specifically, we assume that the loss function $f_t(x)$ takes the form $c_t^\top x$, where

c_t is a bounded loss vector in $\mathcal{C} \subseteq \mathbb{R}^d$. We further assume that the player receives a *hint* before choosing the action on each round. The hint in our setting is a vector that is guaranteed to be weakly correlated with the loss vector. Namely, at the beginning of round t, the player observes a unit-length vector $v_t \in \mathbb{R}^d$ such that $v_t^\top c_t \geq \alpha \|c_t\|_2$, and where α is a small positive constant. So long as this requirement is met, the hint could be chosen maliciously, possibly by an adversary who knows how the player's algorithm uses the hint. Our goal is to develop a player strategy that takes these hints into account, and to understand when hints of this type make the problem provably easier and lead to smaller regret.

We show that the player's ability to benefit from the hints depends on the geometry of the player's action set $\mathcal{K}$. Specifically, we characterize the roundness of the set $\mathcal{K}$ using the notion of (C, q)-uniform convexity for convex sets. In Section 6.4, we show that if $\mathcal{K}$ is a $(C, 2)$-uniformly convex set (or in other words, if $\mathcal{K}$ is a C-strongly convex set), then we can use the hint to design a player strategy that improves the regret guarantee to $O\big((C\alpha)^{-1} \log(T)\big)$, where our $O(\cdot)$ notation hides a polynomial dependence on the dimension d and other constants. Furthermore, as we show in Section 6.5, if $\mathcal{K}$ is a (C, q)-uniformly convex set for $q \in (2, 3)$, we can use the hint to improve the regret to $O\left((C\alpha)^{\frac{1}{1-q}} T^{\frac{2-q}{1-q}}\right)$, when the hint belongs to a small set of possible hints at every step. In Section 6.6, we show that when $\mathcal{K}$ is not uniformly convex, such as an L_1 and L_∞, no algorithm can have a regret of $o(\sqrt{T})$.

6.2 Related work

The notion of hint introduced here is quite general and arises naturally in multiple environments. Indeed, this notion generalizes some of the previous notions of predictability in online optimization. Hazan and Megiddo [157] considered as an example a setting where the player knows the first coordinate of the loss vector at all rounds, and showed that when $c_{t1} > 0$ and the set of feasible actions is the Euclidean ball, one can achieve a regret of $O(\log(T))$. Our work directly improves over this result, as in our setting a hint $v_t = e_1$ also achieves $O(\log(T))$ regret. Furthermore, the algorithm of Hazan and Megiddo [157] requires knowing the value of c_{t1} a priori, whereas our algorithm only needs to know whether $c_{t1} > 0$ and can also work with hints in different directions at the different rounds. Hazan and Megiddo [157] also considered modeling the prior information in each round as a space state and measuring regret against a stronger benchmark that uses a mapping from the state space to the feasible set. Chiang and Lu [78] considered actively querying bits of the loss vector, but their result mainly improves the dependence of regret on the dimension, d.

Another notion of predictability is concerned with predictability of the entire loss vector rather than individual bits. Chiang et al. [79] considered online convex optimization with a sequence of loss functions that demonstrates a gradual change and derived a regret bound in terms of the deviation of the loss functions. Rakhlin and Sridharan [227, 230] extended this line of work beyond sequences with gradual change and showed that one can achieve an improved regret bound if the gradient of the loss function is predictable. They also applied this method to offline optimization problems such as Max Flow and computing the value of a zero-sum game. In the latter case, they showed that when both players employ a variant of the Mirror Prox algorithm, they converge to the minimax equilibrium at rate $O(\log(T))$. In the context of online

linear optimization, our notion of hint generalizes these notions of predictability. We discuss this further in Section 6.7.

6.3 Preliminaries

We begin with a more formal definition of online linear optimization (without hints). Let $\mathcal{A}$ denote the player's algorithm for choosing its actions. On round t the player uses $\mathcal{A}$ and all of the information it has observed so far to choose an action x_t in a convex compact set $\mathcal{K} \subseteq \mathbb{R}^d$. Subsequently, the adversary chooses a loss vector c_t in a compact set $\mathcal{C} \subseteq \mathbb{R}^d$. The player and the adversary reveal their actions and the player incurs the loss $c_t^\top x_t$. The player's regret is defined as

$$\textsc{Regret}(\mathcal{A}, c_{1:T}) \;=\; \sum_{t=1}^{T} c_t^\top x_t - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} c_t^\top x.$$

In online linear optimization *with hints*, the player observes $v_t \in \mathbb{R}^d$ with $\|v_t\|_2 = 1$, before choosing x_t, with the guarantee that $v_t^\top c_t \geq \alpha \|c_t\|_2$, for some $\alpha > 0$.

We use *uniform convexity* to characterize the degree of convexity of the player's action set $\mathcal{K}$. Informally, uniform convexity requires that the convex combination of any two points x and y on the boundary of $\mathcal{K}$ be sufficiently far from the boundary. A formal definition is given below.

Definition 6.3.1 (Pisier [222])**.** *Let $\mathcal{K}$ be a convex set that is symmetric around the origin. $\mathcal{K}$ and the Banach space defined by $\mathcal{K}$ are said to be* uniformly convex *if for any $0 < \epsilon < 2$ there exists a $\delta > 0$ such that for any pair of points $x, y \in \mathcal{K}$ with $\|x\|_{\mathcal{K}} \leq 1, \|y\|_{\mathcal{K}} \leq 1, \|x - y\|_{\mathcal{K}} \geq \epsilon$, we have $\left\|\frac{x+y}{2}\right\|_{\mathcal{K}} \leq 1 - \delta$. The* modulus of uniform-convexity $\delta_{\mathcal{K}}(\epsilon)$ *is the best possible δ for that ϵ, i.e.,*

$$\delta_{\mathcal{K}}(\epsilon) = \inf \left\{ 1 - \left\| \frac{x+y}{2} \right\|_{\mathcal{K}} : \|x\|_{\mathcal{K}} \leq 1, \|y\|_{\mathcal{K}} \leq 1, \|x - y\|_{\mathcal{K}} \geq \epsilon \right\}.$$

For brevity, we say that $\mathcal{K}$ is (C, q)-uniformly convex when $\delta_{\mathcal{K}}(\epsilon) = C\epsilon^q$ and we omit C when it is clear from the context.

Examples of uniformly convex sets include L_p balls for any $1 < p < \infty$ with modulus of convexity $\delta_{L_p}(\epsilon) = C_p \epsilon^p$ for $p \geq 2$ and a constant C_p and $\delta_{L_p}(\epsilon) = (p-1)\epsilon^2$ for $1 < p \leq 2$. On the other hand, L_1 and L_∞ units balls are not uniformly convex. When $\delta_{\mathcal{K}}(\epsilon) \in \Theta(\epsilon^2)$, we say that $\mathcal{K}$ is *strongly convex*.

Another notion of convexity we use in this work is called *exp-concavity*. A function $f : \mathcal{K} \to \mathbb{R}$ is exp-concave with parameter $\beta > 0$, if $\exp(-\beta f(x))$ is a concave function of $x \in \mathcal{K}$. This is a weaker requirement than strong convexity when the gradient of f is uniformly bounded [158]. The next proposition shows that we can obtain regret bounds of order $\Theta(\log(T))$ in online convex optimization when the loss functions are exp-concave with parameter β.

Proposition 6.3.2 (Hazan et al. [158])**.** *Consider online convex optimization on a sequence of loss functions $f_1, \ldots, f_T$ over a feasible set $\mathcal{K} \subseteq \mathbb{R}^d$, such that all t, $f_t : \mathcal{K} \to \mathbb{R}$ is exp-concave with parameter $\beta > 0$. There is an algorithm, with runtime polynomial in d, which we call $\mathcal{A}_{\text{EXP}}$, with a regret that is at most $\frac{d}{\beta}(1 + \log(T+1))$.*

Throughout this work, we draw intuition from basic orthogonal geometry. Given any vector x and a hint v, we define $x^{\parallel v} = (x^\top v)v$ and $x^{\perp v} = x - (x^\top v)v$, as the parallel and the orthogonal components of x with respect to v. When the hint v is clear from the context we simply use $x^\parallel$ and $x^\perp$ to denote these vectors.

Naturally, our regret bounds involve a number of geometric parameters. Since the set of actions of the adversary $\mathcal{C}$ is compact, we can find $G \geq 0$ such that $\|c\|_2 \leq G$ for all $c \in \mathcal{C}$. When $\mathcal{K}$ is uniformly convex, we denote $\mathcal{K} = \{w \in \mathbb{R}^d \mid \|w\|_\mathcal{K} \leq 1\}$. In this case, since all norms are equivalent in finite dimension, there exist $R > 0$ and $r > 0$ such that $B_r \subseteq \mathcal{K} \subseteq B_R$, where B_r (resp. B_R) denote the L_2 unit ball centered at 0 with radius r (resp. R). This implies that $\frac{1}{R}\|\cdot\|_2 \leq \|\cdot\|_\mathcal{K} \leq \frac{1}{r}\|\cdot\|_2$.

6.4 Improved Regret Bounds for Strongly Convex $\mathcal{K}$

At first sight, it is not immediately clear how one should use the hint. Since v_t is guaranteed to satisfy $c_t^\top v_t \geq \alpha\|c_t\|_2$, moving the action x in the direction $-v_t$ always decreases the loss. One could hope to get the most benefit out of the hint by choosing x_t to be the extremal point in $\mathcal{K}$ in the direction $-v_t$. However, this naïve strategy could lead to a linear regret in the worst case. For example, say that $c_t = (1, \frac{1}{2})$ and $v_t = (0, 1)$ for all t and let $\mathcal{K}$ be the Euclidean unit ball. Choosing $x_t = -v_t$ would incur a loss of $-\frac{T}{2}$, while the best fixed action in hindsight, the point $(\frac{-2}{\sqrt{5}}, \frac{-1}{\sqrt{5}})$, would incur a loss of $\frac{-\sqrt{5}}{2}T$. The player's regret would therefore be $\frac{\sqrt{5}-1}{2}T$.

Intuitively, the flaw of this naïve strategy is that the hint does not give the player any information about the $(d-1)$-dimensional subspace orthogonal to v_t. Our solution is to use standard online learning machinery to learn how to act in this orthogonal subspace. Specifically, on round t, we use v_t to define the following *virtual loss function*:

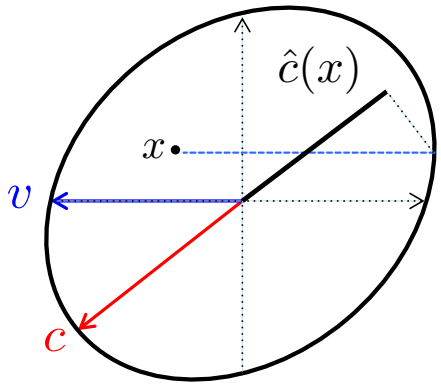
$$\hat{c}_t(x) \;=\; \min_{w \in \mathcal{K}} \; c_t^\top w \quad \text{s.t.} \quad w^{\perp v_t} = x^{\perp v_t} \;\; .$$

In words, we consider the 1-dimensional subspace spanned by v_t and its $(d-1)$-dimensional orthogonal subspace separately. For any action $x \in \mathcal{K}$, we find another point, $w \in \mathcal{K}$, that equals x in the $(d-1)$-dimensional orthogonal subspace, but otherwise incurs the optimal loss. The value of the virtual loss $\hat{c}_t(x)$ is defined to be the value of the original loss function c_t at w. The virtual loss simulates the process of moving x as far as possible in the direction $-v_t$ without changing its value in any other direction (see Figure 6.1a). This can be more formally seen by the following equation.

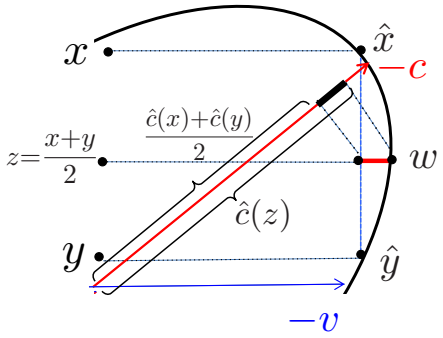
$$\operatorname*{arg\,min}_{w \in \mathcal{K}: w^\perp = \hat{x}^\perp} c_t^\top w = \operatorname*{arg\,min}_{w \in \mathcal{K}: w^\perp = \hat{x}^\perp} \left((c_t^\perp)^\top \hat{x}^\perp + (c_t^\parallel)^\top w^\parallel\right) = \operatorname*{arg\,min}_{w \in \mathcal{K}: w^\perp = \hat{x}^\perp} v_t^\top w, \tag{6.1}$$

where the last transition holds by the fact that $c_t^\parallel = \left\|c_t^\parallel\right\|_2 v_t$ since the hint is valid.

This provides an intuitive understanding of a measure of convexity of our virtual loss functions. When $\mathcal{K}$ is uniformly convex then the function $\hat{c}_t(\cdot)$ demonstrates convexity in the subspace orthogonal to v_t. To see that, note that for any x and y that lie in the space orthogonal to v_t, their mid point $\frac{x+y}{2}$ transforms to a point that is farther away in the direction of $-v_t$ than the midpoint of the transformations of x and y. As shown in Figure 6.1b, the modulus of uniform



(a) Virtual function $\hat{c}(\cdot)$.



(b) Uniform-convexity of the feasible set affects the convexity the virtual loss function.

Figure 6.1: Virtual function and its properties.

convexity of $\mathcal{K}$ affects the degree of convexity of $\hat{c}_t(\cdot)$. We note, however, that $\hat{c}_t(\cdot)$ is not strongly convex in all directions. In fact, $\hat{c}_t(\cdot)$ is constant in the direction of v_t. Nevertheless, the properties shown here allude to the fact that $\hat{c}_t(\cdot)$ demonstrates some notion of convexity. As we show in the next lemma, this notion is indeed *exp-concavity*:

Lemma 6.4.1. *If $\mathcal{K}$ is $(C, 2)$-uniformly convex, then $\hat{c}_t(\cdot)$ is $8\frac{\alpha \cdot C \cdot r}{G \cdot R^2}$-exp-concave.*

Proof. Let $\gamma = 8\frac{\alpha \cdot C \cdot r}{G \cdot R^2}$. Without loss of generality, we assume that $c_t \neq 0$, otherwise $\hat{c}_t(\cdot) = 0$ is a constant function and the proof follows immediately. Based on the above discussion, it is not hard to see that $\hat{c}_t(\cdot)$ is continuous. It remains to prove that $\hat{c}_t(\cdot)$ is exp-concave. For this, it is sufficient to show that

$$\exp\left(-\gamma \cdot \hat{c}_t\left(\frac{x+y}{2}\right)\right) \geq \frac{1}{2}\exp\left(-\gamma \cdot \hat{c}_t(x)\right) + \frac{1}{2}\exp\left(-\gamma \cdot \hat{c}_t(y)\right) \quad \forall (x, y) \in \mathcal{K}.$$

Consider $(x, y) \in \mathcal{K}$ and choose corresponding $(\hat{x}, \hat{y}) \in \mathcal{K}$ such that $\hat{c}_t(x) = c_t^\top \hat{x}$ and $\hat{c}_t(y) = c_t^\top \hat{y}$. Without loss of generality, we have $\|\hat{x}\|_{\mathcal{K}} = \|\hat{y}\|_{\mathcal{K}} = 1$, as we can always choose corresponding $\hat{x}, \hat{y}$ that are extreme points of $\mathcal{K}$. Since $\exp(-\gamma \hat{c}_t(\cdot))$ is decreasing in $\hat{c}_t(\cdot)$, we have

$$\exp\left(-\gamma \cdot \hat{c}_t\left(\frac{x+y}{2}\right)\right) = \max_{\substack{\|w\|_{\mathcal{K}} \leq 1 \\ w^{\perp v_t} = (\frac{x+y}{2})^{\perp v_t}}} \exp(-\gamma \cdot c_t^\top w). \tag{6.2}$$

Note that $w = \frac{\hat{x}+\hat{y}}{2} - \delta_{\mathcal{K}}(\|\hat{x}-\hat{y}\|_{\mathcal{K}})\frac{v_t}{\|v_t\|_{\mathcal{K}}}$ satisfies $\|w\|_{\mathcal{K}} \leq 1$, since $\|w\|_{\mathcal{K}} \leq \left\|\frac{\hat{x}+\hat{y}}{2}\right\|_{\mathcal{K}} + \delta_{\mathcal{K}}(\|\hat{x}-\hat{y}\|_{\mathcal{K}}) \leq 1$ (see also Figure 6.1b). Moreover, $w^{\perp v_t} = (\frac{x+y}{2})^{\perp v_t}$. So, by using this w in Equation (6.2), we have

$$\exp\left(-\gamma \cdot \hat{c}_t\left(\frac{x+y}{2}\right)\right) \geq \exp\left(-\frac{\gamma}{2} \cdot (c_t^\top \hat{x} + c_t^\top \hat{y}) + \gamma \cdot \frac{c_t^\top v_t}{\|v_t\|_{\mathcal{K}}} \cdot \delta_{\mathcal{K}}(\|\hat{x}-\hat{y}\|_{\mathcal{K}})\right). \tag{6.3}$$

On the other hand, since $\|v_t\|_{\mathcal{K}} \leq \frac{1}{r}\|v_t\|_2 = \frac{1}{r}$ and $\|\hat{x}-\hat{y}\|_{\mathcal{K}} \geq \frac{1}{R}\|\hat{x}-\hat{y}\|_2$, we have

$$\begin{aligned}
\exp\left(\gamma \cdot \frac{c_t^\top v_t}{\|v_t\|_{\mathcal{K}}} \cdot \delta_{\mathcal{K}}(\|\hat{x}-\hat{y}\|_{\mathcal{K}})\right) &\geq \exp\left(\gamma \cdot r \cdot \alpha \cdot \|c_t\|_2 \cdot C \cdot \frac{1}{R^2} \cdot \|\hat{x}-\hat{y}\|_2^2\right) \\
&\geq \exp\left(\gamma \cdot \frac{\alpha \cdot C \cdot r}{R^2} \cdot \|c_t\|_2 \cdot \left(\frac{c_t^\top \hat{x}}{\|c_t\|_2} - \frac{c_t^\top \hat{y}}{\|c_t\|_2}\right)^2\right) \\
&\geq \exp\left(\frac{(\gamma/2)^2 \cdot (c_t^\top \hat{x} - c_t^\top \hat{y})^2}{2}\right) \\
&\geq \frac{1}{2} \cdot \exp\left(\frac{\gamma}{2} \cdot (c_t^\top \hat{x} - c_t^\top \hat{y})\right) + \frac{1}{2} \cdot \exp\left(\frac{\gamma}{2} \cdot (c_t^\top \hat{y} - c_t^\top \hat{x})\right),
\end{aligned}$$

where the penultimate inequality follows by the definition of γ and the last inequality is a consequence of the inequality $\exp(z^2/2) \geq \frac{1}{2}\exp(z) + \frac{1}{2}\exp(-z), \forall z \in \mathbb{R}$. Plugging the last inequality into Equation (6.3) yields

$$\begin{aligned}
\exp\left(-\gamma \hat{c}_t(\frac{x+y}{2})\right) &\geq \frac{1}{2}\exp\left(-\frac{\gamma}{2}(c_t^\top \hat{x} + c_t^\top \hat{y})\right) \cdot \left\{\exp\left(\frac{\gamma}{2}(c_t^\top \hat{x} - c_t^\top \hat{y})\right) + \exp\left(\frac{\gamma}{2}(c_t^\top \hat{y} - c_t^\top \hat{x})\right)\right\} \\
&= \frac{1}{2}\exp\left(-\gamma \cdot c_t^\top \hat{y}\right) + \frac{1}{2}\exp\left(-\gamma \cdot c_t^\top \hat{x}\right) \\
&= \frac{1}{2}\exp\left(-\gamma \cdot \hat{c}_t(y)\right) + \frac{1}{2}\exp\left(-\gamma \cdot \hat{c}_t(x)\right),
\end{aligned}$$

which concludes the proof. □

Now, we use the sequence of virtual loss functions to reduce our problem to a standard online convex optimization problem (without hints). Namely, the player applies $\mathcal{A}_{\text{EXP}}$ (from Proposition 6.3.2), which is an online convex optimization algorithm known to have $O(\log(T))$ regret with respect to exp-concave functions, to the sequence of virtual loss functions. Then our algorithm takes the action $\hat{x}_t \in \mathcal{K}$ that is prescribed by $\mathcal{A}_{\text{EXP}}$ and moves it as far as possible in the direction of $-v_t$. This process is formalized in Algorithm 6.1.

Algorithm 6.1: $\mathcal{A}_{\text{hint}}$ FOR STRONGLY CONVEX $\mathcal{K}$

1: **for** $t = 1, \ldots, T$, **do**
2: Use Algorithm $\mathcal{A}_{\text{EXP}}$ with the history $\hat{c}_\tau(\cdot)$ for $\tau < t$, and let $\hat{x}_t$ be the chosen action.
3: Let

$$x_t = \arg\min_{w \in \mathcal{K}} (v_t^\top w) \quad \text{s.t.} \quad w^{\perp v_t} = \hat{x}_t^{\perp v_t}$$

4: Play x_t and receive c_t as feedback.
5: **end for**

Next, we show that the regret of algorithm $\mathcal{A}_{\text{EXP}}$ on the sequence of virtual loss functions is an upper bound on the regret of Algorithm 6.1.

Lemma 6.4.2. *For any sequence of loss functions $c_1, \ldots, c_T$, let* $\textsc{Regret}(\mathcal{A}_{\text{hint}}, c_{1:T})$ *be the regret of algorithm $\mathcal{A}_{\text{hint}}$ on the sequence $c_1, \ldots, c_T$, and* $\textsc{Regret}(\mathcal{A}_{\text{EXP}}, \hat{c}_{1:T})$ *be the regret of algorithm $\mathcal{A}_{\text{EXP}}$ on the sequence of virtual loss functions $\hat{c}_1, \ldots, \hat{c}_T$. Then,* $\textsc{Regret}(\mathcal{A}_{\text{hint}}, c_{1:T}) \leq \textsc{Regret}(\mathcal{A}_{\text{EXP}}, \hat{c}_{1:T})$.

Proof. Equation (6.1) provides an equivalent definition $x_t = \arg\min_{w \in \mathcal{K}}(c_t^\top w)$ s.t. $w^{\perp v_t} = \hat{x}_t^{\perp v_t}$. Using this, we show that the loss of algorithm $\mathcal{A}_{\text{hint}}$ on the sequence $c_{1:T}$ is the same as the loss of algorithm $\mathcal{A}_{\text{EXP}}$ on the sequence $\hat{c}_{1:T}$.

$$\sum_{t=1}^{T} \hat{c}_t(\hat{x}_t) = \sum_{t=1}^{T} \min_{w \in \mathcal{K}: w^\perp = \hat{x}_t^\perp} c_t^\top w = \sum_{t=1}^{T} c_t^\top \Big(\arg\min_{w \in \mathcal{K}: w^\perp = \hat{x}_t^\perp} c_t^\top w \Big) = \sum_{t=1}^{T} c_t^\top x_t.$$

Next, we show that the offline optimal on the sequence $\hat{c}_{1:T}$ is more competitive that the offline optimal on the sequence $c_{1:T}$. First note that for any x and t, $\hat{c}_t(x) = \min_{w \in \mathcal{K}: w^\perp = x^\perp} c_t^\top w \leq c_t^\top x$. Therefore, $\min_{x \in \mathcal{K}} \sum_{t=1}^{T} \hat{c}_t(x) \leq \min_{x \in \mathcal{K}} \sum_{t=1}^{T} c_t^\top x$. The proof concludes by

$$\begin{aligned}
\textsc{Regret}(\mathcal{A}_{\text{hint}}, c_{1:T}) &= \sum_{t=1}^{T} c_t^\top x_t - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} c_t^\top x \\
&\leq \sum_{t=1}^{T} \hat{c}_t(\hat{x}_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} \hat{c}_t(x) \\
&= \textsc{Regret}(\mathcal{A}_{\text{EXP}}, \hat{c}_{1:T}).
\end{aligned}$$

□

Our main result follows from the application of Lemmas 6.4.1 and 6.4.2.

Theorem 6.4.3. *Suppose that $\mathcal{K} \subseteq \mathbb{R}^d$ is a $(C, 2)$-uniformly convex set that is symmetric around the origin, and $B_r \subseteq \mathcal{K} \subseteq B_R$ for some r and R. Consider online linear optimization with hints where the cost function at round t is $\|c_t\|_2 \leq G$ and the hint v_t is such that $c_t^\top v_t \geq \alpha \|c_t\|_2$, while $\|v_t\|_2 = 1$. Algorithm 6.1 in combination with $\mathcal{A}_{\text{EXP}}$ has a worst-case regret of*

$$\textsc{Regret}(\mathcal{A}_{\text{hint}}, c_{1:T}) \leq \frac{d \cdot G \cdot R^2}{8\alpha \cdot C \cdot r} \cdot (1 + \log(T+1)).$$

6.5 Improved Regret Bounds for (C, q)-Uniformly Convex $\mathcal{K}$

In this section, we consider any feasible set $\mathcal{K}$ that is (C, q)-uniformly convex for $q \geq 2$. Our results differ from the previous section in two aspects. First, our algorithm can be used with (C, q)-uniformly convex feasible sets for any $q \geq 2$ compared to the results of the previous section that only hold for strongly convex sets ($q = 2$). On the other hand, the approach in this section requires the hints to be restricted to a finite set of vectors $\mathcal{V}$. We show that when $\mathcal{K}$ is (C, q)-uniformly convex for $q > 2$, our regret is $O(T^{\frac{2-q}{1-q}})$. If $q \in (2, 3)$, this is an improvement over the worst case regret of $O(\sqrt{T})$ guaranteed in the absence of hints.

We first consider the scenario where the hint is always pointing in the same direction, i.e. $v_t = v$ for some v and all $t \in [T]$. In this case, we show how one can use a simple algorithm that picks the best performing action so far (a.k.a the Follow-The-Leader algorithm) to obtain improved regret bounds. We then consider the case where the hint belongs to a finite set $\mathcal{V}$. In this case, we instantiate one copy of the Follow-The-Leader algorithm for each $v \in \mathcal{V}$ and combine their outcomes in order to obtain improved regret bounds that depend on the cardinality of $\mathcal{V}$, which we denote by $|\mathcal{V}|$.

Lemma 6.5.1. *Suppose that $v_t = v$ for all $t = 1, \cdots, T$ and that $\mathcal{K}$ is (C, q)-uniformly convex that is symmetric around the origin, and $B_r \subseteq \mathcal{K} \subseteq B_R$ for some r and R. Consider the algorithm, called Follow-The-Leader (FTL), that at every round t, plays $x_t \in \arg\min_{x \in \mathcal{K}} \sum_{\tau<t} c_\tau^\top x$. If $\sum_{\tau=1}^t c_\tau^\top v \geq 0$ for all $t = 1, \cdots, T$, then the regret is bounded as follows,*

$$\textsc{Regret}(\mathcal{A}_{\text{FTL}}, c_{1:T}) \leq \left(\frac{\|v\|_{\mathcal{K}} \cdot R^q}{2C}\right)^{1/(q-1)} \cdot \sum_{t=1}^{T} \left(\frac{\|c_t\|_2^q}{\sum_{\tau=1}^t c_\tau^\top v}\right)^{1/(q-1)}.$$

Furthermore, when v is a valid hint with margin α, i.e., $c_t^\top v \geq \alpha \cdot \|c_t\|_2$ for all $t = 1, \cdots, T$, the right-hand side can be further simplified to obtain the regret bound:

$$\textsc{Regret}\mathcal{A}_{\text{FTL}}, c_{1:T}) \leq \frac{1}{2\gamma} \cdot G \cdot (\ln(T) + 1) \qquad \text{if } q = 2$$

and

$$\textsc{Regret}(\mathcal{A}_{\text{FTL}}, c_{1:T}) \leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot G \cdot \frac{q-1}{q-2} \cdot T^{\frac{q-2}{q-1}} \qquad \text{if } q > 2,$$

where $\gamma = \frac{C \cdot \alpha}{\|v\|_{\mathcal{K}} \cdot R^q}$.

Proof. We use a well-known inequality, known as FT(R)L Lemma (see e.g., [154, 208]), on the regret incurred by the FTL algorithm:

$$\textsc{Regret}(\mathcal{A}_{\text{FTL}}, c_{1:T}) \leq \sum_{t=1}^{T} c_t^\top (x_t - x_{t+1}).$$

Without loss of generality, we can assume that $\|x_t\|_{\mathcal{K}} = \|x_{t+1}\|_{\mathcal{K}} = 1$ since the maximum of a linear function is attained at a boundary point. Since $\mathcal{K}$ is (C, q)-uniformly convex, we have

$$\left\|\frac{x_t + x_{t+1}}{2}\right\|_{\mathcal{K}} \leq 1 - \delta_{\mathcal{K}}(\|x_t - x_{t+1}\|_{\mathcal{K}}).$$

This implies that

$$\left\|\frac{x_t + x_{t+1}}{2} - \delta_{\mathcal{K}}(\|x_t - x_{t+1}\|_{\mathcal{K}}) \frac{v}{\|v\|_{\mathcal{K}}}\right\|_{\mathcal{K}} \leq 1.$$

Moreover, $x_{t+1} \in \arg\min_{x \in \mathcal{K}} x^\top \sum_{\tau=1}^t c_\tau$. So, we have

$$\left(\sum_{\tau=1}^{t} c_\tau\right)^\top \left(\frac{x_t + x_{t+1}}{2} - \delta_{\mathcal{K}}(\|x_t - x_{t+1}\|_{\mathcal{K}}) \frac{v}{\|v\|_{\mathcal{K}}}\right) \geq \inf_{x \in \mathcal{K}} x^\top \sum_{\tau=1}^{t} c_\tau = x_{t+1}^\top \sum_{\tau=1}^{t} c_\tau.$$

Rearranging this last inequality and using the fact that $\sum_{\tau=1}^{t} v^\top c_\tau \geq 0$, we obtain:

$$\left(\sum_{\tau=1}^{t} c_\tau\right)^\top \left(\frac{x_t - x_{t+1}}{2}\right) \geq \delta_{\mathcal{K}}(\|x_t - x_{t+1}\|_{\mathcal{K}}) \cdot \frac{\sum_{\tau=1}^{t} v^\top c_\tau}{\|v\|_{\mathcal{K}}} \geq \frac{C \cdot \|x_t - x_{t+1}\|_2^q}{\|v\|_{\mathcal{K}} \cdot R^q} \cdot \left(\sum_{\tau=1}^{t} v^\top c_\tau\right).$$

By definition of FTL, we have $x_t \in \arg\min_{x \in \mathcal{K}} x^\top \sum_{\tau=1}^{t-1} c_\tau$, which implies:

$$\left(\sum_{\tau=1}^{t-1} c_\tau\right)^\top \frac{x_{t+1} - x_t}{2} \geq 0.$$

Summing up the last two inequalities, we derive:

$$\begin{aligned} c_t^\top \left(\frac{x_t - x_{t+1}}{2}\right) &\geq \left(\frac{C}{\|v\|_{\mathcal{K}} \cdot R^q}\right) \cdot \left(\sum_{\tau=1}^{t} v^\top c_\tau\right) \cdot \|x_t - x_{t+1}\|_2^q \\ &\geq \left(\frac{C}{\|v\|_{\mathcal{K}} \cdot R^q}\right) \cdot \left(\sum_{\tau=1}^{t} v^\top c_\tau\right) \cdot \frac{(c_t^\top (x_t - x_{t+1}))^q}{\|c_t\|_2^q}. \end{aligned}$$

Rearranging this last inequality and using the fact that $\sum_{\tau=1}^{t} v^\top c_\tau \geq 0$, we obtain:

$$|c_t^\top (x_t - x_{t+1})| \leq \left(\frac{\|v\|_{\mathcal{K}} \cdot R^q}{2C}\right)^{1/(q-1)} \cdot \left(\frac{\|c_t\|_2^q}{\sum_{\tau=1}^{t} v^\top c_\tau}\right)^{1/(q-1)}. \tag{6.4}$$

Summing Equation (6.4) over all t completes the proof of the first claim.

The regret bounds for when $v^\top c_t \geq \alpha \cdot \|c_t\|_2$ for all $t = 1, \cdots, T$ follow from the first regret bound and setting $\gamma = \frac{C \cdot \alpha}{\|v\|_{\mathcal{K}} \cdot R^q}$, we have:

$$|c_t^\top (x_t - x_{t+1})| \leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot \left(\frac{\|c_t\|_2^q}{\sum_{\tau=1}^{t} \|c_\tau\|_2}\right)^{1/(q-1)}.$$

Note that the right-hand side is finite even if $c_t = 0$. Plugging this last inequality back into the first regret bound, we derive:

$$\begin{aligned} \textsc{Regret}(\mathcal{A}_{\text{FTL}}, c_{1:T}) &\leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot \sum_{t=1}^{T} \left(\frac{\|c_t\|_2^q}{\sum_{\tau=1}^{t} \|c_\tau\|_2}\right)^{1/(q-1)} \\ &\leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot \sup_{(y_1, \cdots, y_T) \in [0,G]^T} \sum_{t=1}^{T} \left(\frac{y_t^q}{\sum_{\tau=1}^{t} y_\tau}\right)^{1/(q-1)}. \end{aligned}$$

We prove below that for any $t = 1, \cdots, T$ and any fixed values $(y_1, \cdots, y_{t-1}, y_{t+1}, \cdots, y_T) \in [0, G]^{T-1}$, the function $y_t \to \sum_{n=1}^{T} \left(\frac{y_n^q}{\sum_{\tau=1}^{n} y_\tau}\right)^{1/(q-1)}$ is convex on $[0, G]$ and thus the maximum

of this function is attained at an extreme point: either 0 or G. Repeating this process for $t = 1, \cdots, T$, we get:

$$\begin{aligned}
\textsc{Regret}(\mathcal{A}_{\text{FTL}}, c_{1:T}) &\leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot \sup_{(y_1,\cdots,y_T)\in\{0,G\}^T} \sum_{t=1}^{T} \left(\frac{y_t^q}{\sum_{\tau=1}^{t} y_\tau} \right)^{1/(q-1)} \\
&\leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot \sup_{n=0,\cdots,T} \sum_{k=1}^{n} (\frac{G^q}{k \cdot G})^{1/(q-1)} \\
&\leq \frac{1}{(2\gamma)^{1/(q-1)}} \cdot G \cdot \sum_{k=1}^{T} \frac{1}{k^{1/(q-1)}}.
\end{aligned}$$

This concludes the proof as $\sum_{k=1}^{T} \frac{1}{k^{1/(q-1)}} \leq \ln(T) + 1$ if $q = 2$ and $\sum_{k=1}^{T} \frac{1}{k^{1/(q-1)}} \leq \frac{q-1}{q-2} \cdot T^{\frac{q-2}{q-1}}$ if $q > 2$.

Let us now prove that, for any $t = 1, \cdots, T$ and any fixed values $(y_1, \cdots, y_{t-1}, y_{t+1}, \cdots, y_T) \in [0, G]^{T-1}$, the function $y_t \to \sum_{n=1}^{T} (\frac{y_n^q}{\sum_{\tau=1}^{n} y_\tau})^{1/(q-1)}$ is convex on $[0, G]$. Clearly,

$$y_t \to \sum_{n\neq t} (\frac{y_n^q}{\sum_{\tau=1}^{n} y_\tau})^{1/(q-1)}$$

is convex on $[0, G]$ since y_t only appears in the denominator and $1/(q-1) \geq 0$. We use the shorthand $A = \sum_{\tau=1}^{t-1} y_\tau$. It remains to show that $\phi : y \to (\frac{y^q}{y+A})^{1/(q-1)}$ is convex. We have:

$$\phi''(y) = \frac{q}{(q-1)^2} \cdot y^{q/(q-1)-2} \cdot A^2 \cdot (y + A)^{-1/(q-1)-2},$$

which is non-negative for $A \geq 0$ and $y > 0$ (for $y = 0$, we can directly show by hand that $\phi(\lambda \cdot 0 + (1-\lambda) \cdot z) \leq \lambda \cdot \phi(0) + (1-\lambda) \cdot \phi(z)$ for $\lambda \in [0, 1]$ and $z \geq 0$).

□

Note that the regret bounds become $O(T)$ when $q \to \infty$. This is expected because L_q balls are q-uniformly convex for $q \geq 2$ and converge to L_∞ balls as $q \to \infty$ and it is well-known that Follow-The-Leader yields $\Theta(T)$ regret in online linear optimization when $\mathcal{K}$ is a L_∞ ball.

Using the above lemma, we introduce an algorithm for online linear optimization with hints that belong to a set $\mathcal{V}$. In this algorithm, we instantiate one copy of the FTL algorithm for each possible direction of the hint. On round t, we invoke the copy of the algorithm that corresponds to the direction of the hint v_t, using the history of the game for rounds with hints in that direction. We show that the overall regret of this algorithm is no larger than the sum of the regrets of the individual copies.

Theorem 6.5.2. *Suppose that $\mathcal{K} \subseteq \mathbb{R}^d$ is a (C, q)-uniformly convex set that is symmetric around the origin, and $B_r \subseteq \mathcal{K} \subseteq B_R$ for some r and R. Consider online linear optimization with hints where the cost function at round t is $\|c_t\|_2 \leq G$ and the hint v_t comes from a finite set $\mathcal{V}$ and is such that $c_t^\top v_t \geq \alpha \|c_t\|_2$, while $\|v_t\|_2 = 1$. Algorithm 6.2 has a worst-case regret of*

$$\textsc{Regret}(\mathcal{A}_{\text{set}}, c_{1:T}) \leq |\mathcal{V}| \cdot \frac{R^2}{2C \cdot \alpha \cdot r} \cdot G \cdot (\ln(T) + 1), \qquad \text{if } q = 2,$$

Algorithm 6.2: $\mathcal{A}_{\text{set}}$: SET-OF-HINTS

1: **for** $v \in \mathcal{V}$, let $T_v = \emptyset$. **end for**
2: **for** $t = 1, \ldots, T$, **do**
3: Play

$$x_t \in \arg\min_{x \in \mathcal{K}} \sum_{\tau \in T_{v_t}} c_\tau^\top x$$

4: Receive c_t as feedback.
5: Update $T_{v_t} \leftarrow T_{v_t} \cup \{t\}$.
6: **end for**

and

$$\text{REGRET}(\mathcal{A}_{\text{set}}, c_{1:T}) \leq |\mathcal{V}| \cdot \left(\frac{R^q}{2C \cdot \alpha \cdot r}\right)^{1/(q-1)} \cdot G \cdot \frac{q-1}{q-2} \cdot T^{\frac{q-2}{q-1}} \qquad \text{if } q > 2.$$

Proof. We decompose the regret as follows:

$$\begin{aligned}\text{REGRET}(\mathcal{A}_{\text{set}}, c_{1:T}) &= \sum_{t=1}^{T} c_t^\top x_t - \inf_{x \in \mathcal{K}} \sum_{t=1}^{T} c_t^\top x \leq \sum_{v \in \mathcal{V}} \left\{ \sum_{t \in T_v} c_t^\top x_t - \inf_{x \in \mathcal{K}} \sum_{t \in T_v} c_t^\top x \right\} \\ &\leq |\mathcal{V}| \cdot \max_{v \in \mathcal{V}} \text{REGRET}(\mathcal{A}_{\text{FTL}}, c_{T_v}).\end{aligned}$$

The proof follows by applying Lemma 6.5.1 and by using $\|v_t\|_{\mathcal{K}} \leq (1/r) \cdot \|v_t\|_2 = 1/r$. □

Note that $\mathcal{A}_{\text{set}}$ does not require α or $\mathcal{V}$ to be known a priori, as it can compile the set of hint directions as it sees new ones. Moreover, if the hints are not limited to finite set $\mathcal{V}$ a priori, then the algorithm can first discretize the L_2 unit ball with an $\alpha/2$-net and approximate any given hint with one of the hints in the discretized set. Using this discretization technique, Theorem 6.5.2 can be extended to the setting where the hints are not constrained to a finite set while having a regret that is linear in the size of the $\alpha/2$-net (exponential in the dimension d).

6.6 Lack of uniform Convexity

In the previous section, we showed that when the feasible set is sufficiently uniformly convex, one can use hints to improve over the worst case regret bound of $\Omega(\sqrt{T})$. In this section, we examine two commonly used feasible sets that are not uniformly convex and show that in these cases even a stronger form of hint cannot guarantee an improved regret bounds.

Consider the feasible set defined by the unit cube, i.e. $\mathcal{K} = \{x \mid \|x\|_\infty \leq 1\}$. Note that this set is not uniformly convex. Since, the i^{th} coordinate of points in such a set, namely x_i, has no effect on the range of acceptable values for the other coordinates, revealing one coordinate does not give us any information about the other coordinates x_j for $j \neq i$. For example, suppose

that c_t has each of its first two coordinates set to $+1$ or -1 with equal probability and all other coordinates set to 1. In this case, even after observing the last $d-2$ coordinates of the loss vector, the problem is reduced to a standard online linear optimization problem in the 2-dimensional unit cube. This choice of c_t is known to incur a regret of $\Omega(\sqrt{T})$ [2]. Therefore, online linear optimization with the set $\mathcal{K} = \{x \mid \|x\|_\infty \leq 1\}$, even in the presence of hints, has a worst-case regret of $\Omega(\sqrt{T})$.

Now consider $\mathcal{K} = \{x \mid \|x\|_1 \leq 1\}$ that is not uniformly convex. As opposed to the unit L_∞ ball, when we fix the value of x_i in the L_1 unit ball we directly affect the range of the acceptable values for x_j such that $\sum_{j \neq i} x_j \leq 1 - |x_i|$. At a high level, revealing the loss on coordinate i when it is the same as the expected loss on the remaining coordinates combined, does not guide the player's actions on the remaining coordinates. The next theorem formalizes this intuition and shows that no online linear optimization algorithm has a worst case regret of $o(\sqrt{T})$ even when it receives hints.

Theorem 6.6.1. *In online linear optimization with hints where the feasible region is $\mathcal{K} = \{x \mid \|x\|_1 = 1\}$, there exists a set of hints $v_1, \ldots, v_t$ such that $c_t^\top v_t \geq \frac{1}{2\sqrt{d}} \|c_t\|$, $\|v_t\| = 1$ for all t, but every online algorithm has worst case regret of $\Omega(\sqrt{T \log d})$.*

Proof. Consider the online linear optimization problem, where the hint at every step is $v_t = (0, 0, \ldots, -1)$. Additionally the adversary's cost function at round t, c_t, is such that $c_{t,i}$ is equal to 0 or -2 with equal probability for $i < d$, and $c_{t,n} = -1$. Note, the hint at round t satisfies the criteria that $\|v_t\|_2 = 1$ and $v_t^\top c_t = 1$. Since $1 \leq \|c_t\| \leq 2\sqrt{d}$, we have that $v_t^\top c_t \geq \left(\frac{1}{2\sqrt{d}}\right) \|c_t\|$, i.e., we have a $\frac{1}{2\sqrt{d}}$-hint. We show that the worst case regret of any algorithm on such a stochastically generated sequence is $\Omega(\sqrt{T \log d})$. First note that even after revealing that $c_{t,n} = -1$ for all t, the expected loss of any online algorithm is in the best case $-T$. This is because the expected loss the player receives is $\sum_{t=1}^{T} \sum_{i=1}^{d} x_i \, \mathbb{E}[c_{t,i}] = -\sum_{t=1}^{T} \sum_{i=1}^{d} x_i \geq -T$.

Now consider the expected loss of the optimal offline solutions. Let the value of the offline optimal solution be OPT. Then $OPT = \min_{\|x\|_1 = 1} \sum_{t=1}^{T} c_t^\top x = -\max_{\|x\|_1 = 1} \sum_{t=1}^{T} -c_t^\top x = -\left\|\sum_{t=1}^{T} c_t\right\|_\infty$, by definition of dual norms. Since $c_{t,i}$ is a binomial variable, for large enough T, $\sum_{t=1}^{T} c_{t,i}$ is approximately a normal variable with mean $-T$ and variance T. Moreover, $\left\|\sum_{t=1}^{T} c_t\right\|_\infty$ is the maximum of $d-1$ independent such normal variables and also a constant -1, therefore $\mathbb{E}\left[\left\|-\sum_{t=1}^{T} c_t\right\|_\infty\right] = -T - \Theta(\sqrt{T \log(d)})$. Hence, the expected regret of any online algorithm is at best $\Theta(\sqrt{T \log(d)})$. □

At first sight, this result may come as a surprise. After all, since any L_p ball with $1 < p \leq 2$ is $\Theta(\epsilon^2)$-uniformly convex, one hopes to use a $L_{1+\nu}$ unit ball K' to approximate K and apply the results of Section 6.4 to achieve better regret bounds. The problem with this approach is that as $p \to 1$, the constant in the modulus of convexity of the the set deteriorates as $\delta_{L_p}(\epsilon) = (p-1)\epsilon^2$ [41]. Therefore, the guarantees Theorem 6.4.3, which is $O\left(\frac{1}{p-1} \log T\right)$ also worsens as $p \to 1$. On the other hand, for p that is significantly larger than 1, $\mathcal{K}'$ becomes a smaller subset of $\mathcal{K}$ and the approximation ratio weakens. Since the best approximation of L_1 ball using an

L_p ball is of the form $\left\{x \mid d^{1-\frac{1}{p}}\|x\|_p \leq 1\right\}$, the distance between the offline optimal in K and $\mathcal{K}'$ can be as large as $1 - d^{1-\frac{1}{p}}$. This adds in an additional regret of $\left(1 - d^{1-\frac{1}{p}}\right) T$. Due to the inverse dependence of these two regret terms on p, the optimal choice of $p = 1 + \tilde{O}\left(\frac{1}{\sqrt{T}}\right)$ leads to regret $\tilde{O}\left(\sqrt{T}\right)$.

6.7 Discussion

In this chapter, we introduced a general model of online linear optimization where on every round the learner has access to a linear function that is weakly correlated with the loss function. We showed that using this hint one can achieve an improved regret bound of $o(\sqrt{T})$ depending on the degree of convexity of the set of feasible action.

6.7.1 Comparison with other Notions of Hint

The notion of hint that we introduce in this chapter generalizes some of the notions of predictability in online learning. Hazan and Megiddo [157] considered as an example a setting where the player knows the first coordinate of the loss vector at all rounds, and showed that when $|c_{t1}| \geq \alpha$ and when the set of feasible actions is the Euclidean ball, one can achieve a regret of $O(1/\alpha \cdot \log(T))$. Our work directly improves over this result, as in our setting a hint $v_t = \pm e_1$ also achieves $O(1/\alpha \cdot \log(T))$ regret, but we can deal with hints in different directions at different rounds and we allow for general uniformly convex action sets.

Rakhlin and Sridharan [230] considered online learning with predictable sequences, with a notion of predictability that is concerned with the gradient of the convex loss functions. They show that if the player receives a hint M_t at round t, then the regret of the algorithm is at most $O(\sqrt{\sum_{t=1}^{T} \|\nabla f_t(x_t) - M_t\|_*^2})$. In the case of linear loss functions, this implies that having an estimate vector c_t' of the loss vector within distance σ of the true loss vector c_t results in an improved regret bound of $O(\sigma\sqrt{T})$. In contrast, we consider a notion of hint that pertains to the *direction of the loss vector* rather than its location. Our work shows that merely knowing whether the loss vector positively or negatively correlates with another vector is sufficient to achieve improved regret bound, when the set is uniformly convex. That is, rather than having access to an approximate value of c_t, we only need to have access to a halfspace that classifies c_t correctly with a margin. This notion of hint is weaker that the notion of hint in the work of Rakhlin and Sridharan [230] when the approximation error satisfies $\|c_t - c_t'\|_2 \leq \sigma \cdot \|c_t\|_2$ for $\sigma \in [0, 1)$. In this case one can use $c_t' / \|c_t'\|_2$ as the direction of the hint in our setting and achieve a regret of $O(\frac{1}{1-\sigma} \log T)$ when the set is strongly convex. This shows that when the set of feasible actions is strongly convex, a directional hint can improve the regret bound beyond what has been known to be achievable by an approximation hint. However, we note that our results require the hints to be always valid, whereas the algorithm of Rakhlin and Sridharan [227] can adapt to the quality of the hints.

An interesting open problem is whether a combination of natural notions of hint can lead to exponential improvement over the regret bound of $O(\sqrt{T})$ even when $\mathcal{K}$ is not uniformly

convex.

Chapter 7

Smoothed Analysis of Online Learning

7.1 Introduction

In this chapter, we continue our examination of a suitable middle ground between *offline* and *online learnability*. Offline and online learnability are two of the most classical problems in the theory of machine learning. The first considers scenarios where a learner attempts to learn a highly accurate hypothesis on instances that are drawn i.i.d. from some fixed but unknown distribution and the second considers online scenarios where the instances are generated by an adversary. It is well-known that offline and online learnability are characterized by two notions of complexity of the hypothesis space: the VC dimension [264] and the Littlestone dimension [193], respectively. In many hypothesis classes, however, there is a large gap between these two notions of complexity, and as a result, there is a gap in our ability to learn in the offline and online settings. For example, it is well known that the class of 1-dimensional threshold functions has a VC dimension of 1 and can be learned in the offline i.i.d. setting with convergence rate of $O\left(\frac{1}{\sqrt{T}}\right)$, but the Littlestone dimension of this class is unbounded, and so learning in the online adversarial setting is impossible. Is there a middle ground between the i.i.d. and adversarial models that can lead to strong learnability results, like those achievable in the offline setting, but is still robust to the presence of an adversary? *Smoothed analysis* provides one approach to this question.

7.1.1 Smoothed Analysis

Smoothed analysis [252] is one of the most well-studied examples of a semi-random model, where nature and an adversary collaborate to produce an input to an algorithm. The idea is that the adversary first chooses a worst-case input, which is then perturbed slightly by nature. Equivalently, an adversary is forced to choose an input distribution that is not overly concentrated, and the input is then drawn from the adversary's chosen distribution. The goal is then to design an algorithm that always (no matter what the adversary does) has good expected performance, where the expectation is over nature's perturbation.

Smoothed analysis can escape the curse of worst-case inputs (especially if they are "brittle"), while also avoiding overfitting a solution to a specific distributional assumption. There is also a

plausible narrative about why "real-world" environments are captured by this framework: even in a world that is out to get us, there are inevitable inaccuracies such as measurement error and uncertainties that *smooth* the environment.

7.1.2 Smoothed Analysis in Online Learning

This chapter extends the reach of smoothed analysis to *regret-minimization in online learning*. We consider the standard online learning setup in which there is an input space $\mathcal{X}$ (assumed to be a bounded subset of Euclidean space), a set $\mathcal{H}$ of binary hypotheses (each mapping $\mathcal{X}$ to $\{+1, -1\}$), and a known time horizon T. Each time step $t = 1, 2, \ldots, T$, an online algorithm chooses a distribution q_t over hypotheses from $\mathcal{H}$, and an adversary subsequently chooses an input point $x_t \in \mathcal{X}$ and a label $y_t \in \{+1, -1\}$. Recall that the *average regret* incurred by a sequence of hypotheses $h_1, \ldots, h_T$ on a sequence of input-label pairs $(x_1, y_1), \ldots, (x_T, y_T)$ is the difference between the average error of $h_1, \ldots, h_T$ (i.e., the fraction of misclassified points) and that of the best fixed hypothesis from $\mathcal{H}$. The goal of an online leaning algorithm is to drive the expected average regret to 0 as quickly as possible (no matter what the adversary does).

We consider *σ-smooth adversaries*, which at each time step t choose an arbitrary distribution $\mathcal{D}_t$ over input-label pairs with a density function over inputs that is pointwise at most $1/\sigma$ times that of the uniform distribution. The actual input at time t is chosen at random according to $\mathcal{D}_t$. The adversary is allowed to set y_t arbitrarily, after observing the realization of x_t. The parameter σ smoothly interpolates between a worst-case adversary (when $\sigma \to 0$) and a uniformly random adversary (when $\sigma = 1$).

Why should smoothed analysis help in online learning? To answer this question, we next review a well known but "brittle" bad example. Suppose $\mathcal{X} = [0, 1]$ and $\mathcal{H}$ is the set of *1-D threshold functions*, where each $h \in \mathcal{H}$ has the form $h(x) = 1$ for all $x \leq b$ and $h(x) = -1$ for all $x > b$ (for some $b \in [0, 1]$). Consider a randomized adversary that always chooses the labels y_t uniformly at random, and the x_t's as follows: x_1 is $\frac{1}{2}$; x_2 is either $\frac{1}{4}$ (if $y_1 = -1$) or $\frac{3}{4}$ (if $y_1 = 1$); and so on. With probability 1, in hindsight, there is a hypothesis with zero error. But every online learning algorithm has a 50% chance of an error at every time step.

This bad example for 1-D thresholds exploits the adversary's ability to specify input points with arbitrary precision. It is intuitively clear (and not very hard to prove) that, with a σ-smooth adversary in this example, it is possible to achieve small expected regret. Thus, online learning can be fundamentally easier with a σ-smooth adversary (even for very small σ) than a worst-case adversary. But how general is this phenomenon?

> Is there an online algorithm with expected regret at most $f(d, \sigma) \cdot T^{-c}$ for every σ-smooth adversary, where d is the VC dimension of $\mathcal{H}$, $f(\cdot, \cdot)$ is an arbitrary function, and $c > 0$ is a constant?

We answer this question for a non-adaptive adversary that specifies all T σ-smooth distributions, $\mathcal{D}_1, \ldots, \mathcal{D}_T$ in advance. We will also discuss the case of the adaptive adversary that can choose $\mathcal{D}_t$ as a function of the history of the play, including the realization of earlier instances and the earlier hypotheses used by the algorithm. Such an adaptive smooth adversary is more powerful than a smoothed non-adaptive one in two senses: it can condition its output on the

outcome of both the *algorithm's* coin flips in previous iterations, and also its *own* past coin flips. Only the first source of power plays a role in the classical online learning setting.

7.1.3 Our Results

The next theorem, whose proof appears in Section 7.3, upper bounds the regret against a non-adaptive smooth adversary.

Theorem 7.3.4 (Informal) *There is an algorithm with expected average regret of* $O\left(\sqrt{\frac{d\ln(T/\sigma)}{T}}\right)$ *against any non-adaptive* σ*-smooth adversary.*

As we showed earlier, there is an adaptive (non-smooth) adversary against whom any online algorithm incurs an average regret of $\Omega(1)$. The next theorem, whose proof appears in Section 7.4 uses a variation of the same example to show that there is also a non-adaptive (and non-smooth) adversary that enforces a regret of $\Omega(1)$ for every online algorithm. Therefore, what enables our strong regret bound in Theorem 7.3.4 is indeed the smoothness of the adversary.

Theorem 7.4.1 (Informal) *There is a non-adaptive (and non-smooth) adversary and a class of hypotheses* $\mathcal{H}$ *with VC dimension* 1*, against which any online algorithm incurs an average regret of* $\Omega(1)$ *as* $T \to \infty$*.*

The above theorems demonstrate that finite VC dimension of a class of hypotheses provides an upper bound of $\tilde{O}\left(\sqrt{d/T}\right)$ on the regret of online algorithms against non-adaptive smooth adversaries. It is worth mentioning that VC dimension also provides a lower bound of $\tilde{\Omega}\left(\sqrt{d/T}\right)$ on the regret of online and offline algorithms (see e.g. [11]). Therefore, the VC dimension of a class of hypotheses characterizes regret against non-adaptive smooth adversaries.

7.1.4 Related Work

To date, most applications of the smoothed analysis concern the computational complexity of algorithms. Here, we discuss a few works that, like ours, have considered smoothed analysis for the purpose of improving the statistical aspects of an algorithm.

The use of smoothed analysis in online learning was first explored by Rakhlin et al. [229] who showed that, under additive perturbations in the instance space, one can achieve $O(\ln(1/\sigma)T^{-1/2})$ regret against the best 1-dimensional halfspace, even though the Littlestone dimension of this hypothesis class is known to be unbounded. Gupta and Roughgarden [138] considered online optimization of Maximum Weighted Independent Set (MWIS) under the smoothed analysis model and showed that while online optimization of MWIS for worst case adversaries leads to a constant regret, against a smoothed adversary one can guarantee a small regret. Subsequently, Cohen-Addad and Kanade [81] studied online optimization of piecewise constant functions under the assumption that the thresholds at which the function changes its value are perturbed and further improved the runtime analysis of the aforementioned result. In comparison, our work considers all hypothesis classes with bounded VC dimension and shows that it is precisely the boundedness of their VC dimension together with the smoothness of the adversary that characterizes their regret bound.

More recently, Kannan et al. [172] considered the performance of the greedy algorithm in the linear contextual bandit problem with stochastic rewards and showed that while the greedy algorithm does not have a good worst-case regret bound, it has a small regret when faced with a smooth adversary. This differs from our line of inquiry in two ways: We focus on characterizing the regret achievable using *any* algorithm while they focus on the regret of the greedy algorithm (for fairness purposes)—indeed, there are non-greedy methods with good regret in their setting. Moreover, the rewards in our setting are adversarial and will be revealed fully after each round. In contrast, in their setting these rewards are stochastic and the learner only receives bandit feedback.

7.2 Preliminaries

Consider a bounded and Lebesgue-measurable instance space $\mathcal{X}$ and the label set $\mathcal{Y} = \{+1, -1\}$. We consider a hypothesis class $\mathcal{H}$. For a labeled instance $s = (x, y)$ and a hypothesis $h \in \mathcal{H}$, $\operatorname{err}_s(h) = \mathbb{1}(h(x) \neq y)$ indicates whether h makes a mistake on s. For a multi-set of samples $\mathbf{s} = (s_1, \ldots, s_T)$, let $\operatorname{err}_{\mathbf{s}}(h) = \frac{1}{T}\sum_{i\in[T]} \operatorname{err}_{s_i}(h)$ denote the empirical error of h.

Let $\mathcal{U}$ be the uniform distribution over $\mathcal{X}$ with density (or mass) function $u(\cdot)$. For a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, let $p(\cdot)$ be the *probability density (or mass) function* of its marginal over $\mathcal{X}$. $\mathcal{D}$ is said to be σ*-smooth* if for all $x \in \mathcal{X}$, $p(x) \leq u(x)\sigma^{-1}$.

In this chapter, we consider the setting of *online (adversarial and full-information) learning*. In this setting, a learner and an adversary interact over T time steps. Our main result is concerned with a *non-adaptive* σ-smooth adversary: In this setting, the adversary first chooses an unknown sequence of distributions $\boldsymbol{\mathcal{D}} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$ such that $\mathcal{D}_t$ is a σ-smooth distribution over $\mathcal{X} \times \mathcal{Y}$ for all $t \in [T]$. At every step $t \in [T]$, the learner picks one $h_t \in \mathcal{H}$. The learner then observes an instance $s_t \sim \mathcal{D}_t$ and incurs penalty $\operatorname{err}_{s_t}(h_t)$. We denote this random process by $\mathbf{s} \sim \boldsymbol{\mathcal{D}}$ and denote by $\operatorname{err}_{\boldsymbol{\mathcal{D}}}(h) = \mathbb{E}_{\mathbf{s}\sim\boldsymbol{\mathcal{D}}}[\operatorname{err}_{\mathbf{s}}(h)]$ the *true error* of hypothesis h on $\boldsymbol{\mathcal{D}}$. We are interested in minimizing the expected average regret,

$$\mathbb{E}[\textsc{Average-Regret}] := \mathop{\mathbb{E}}_{\mathbf{s}}\left[\operatorname*{err}_{\mathbf{s}}(h_t) - \inf_{h\in\mathcal{H}} \operatorname*{err}_{\mathbf{s}}(h)\right],$$

where the expectation is taken over the randomness of the learner and $\mathbf{s} \sim \boldsymbol{\mathcal{D}}$. We desire algorithms, called *no-regret* algorithms, for which this average regret tends to 0 as $T \to \infty$.

Given a hypothesis class $\mathcal{H}$ and a set of m unlabeled instances $\mathbf{x} = (x_1, \ldots, x_m)$, we define $\mathcal{H}[\mathbf{x}] = \{(h(x_1), \ldots, h(x_m))\}_{\forall h\in\mathcal{H}}$ to be the set of all possible ways that hypotheses in $\mathcal{H}$ can classify instances in $\mathbf{x}$. The *growth function* of $\mathcal{H}$, denoted by $\pi_{\mathcal{H}} : \mathbb{N} \to \mathbb{N}$ is defined as

$$\pi_{\mathcal{H}}(m) = \max_{\mathbf{x}\in\mathcal{X}^m} |\mathcal{H}[\mathbf{x}]|.$$

The *VC dimension* of $\mathcal{H}$ is defined as the largest m for which $\pi_{\mathcal{H}}(m) = 2^m$. It is well-known that $\pi_H(m) \leq (em/d)^d$ for all $m \geq d$. In this work, we consider a hypothesis class $\mathcal{H}$ that has a finite VC dimension d.

7.3 Main Results

In this section, we develop fundamental tools for leveraging smoothness of an adversary in an online learning setting.

For finite hypothesis classes $\mathcal{H}$, existing no-regret algorithms such as Hedge [124], Multiplicative Weight Update [196], and Follow-the-Perturbed-Leader [169] provide a regret bound of $O\left(\sqrt{\log(|\mathcal{H}|)/T}\right)$. For a (possibly infinite) hypothesis class $\mathcal{H}$ with finite VC dimension, we seek to use a finite set $\mathcal{H}' \subseteq \mathcal{H}$ as a proxy for $\mathcal{H}$ and only focus on competing with hypotheses in $\mathcal{H}'$. The inexact nature of such an approximation of $\mathcal{H}$ by $\mathcal{H}'$ generally leads to additional regret. The following lemma shows the trade-off between our gain in using a small hypothesis class and the additional loss caused by this approximation for any (smooth or non-smooth) adversary.

Lemma 7.3.1. *Let $\mathcal{H}' \subseteq \mathcal{H}$. Then, there is an algorithm for which*

$$\mathbb{E}[\textsc{Average-Regret}] \leq O\left(\sqrt{\frac{\ln(|\mathcal{H}'|)}{T}}\right) + \frac{1}{T}\mathop{\mathbb{E}}_{\mathbf{s}}\left[\sup_{h\in\mathcal{H}}\inf_{h\in\mathcal{H}'}\sum_{t=1}^{T}\mathbb{1}(h'(s_t)\neq h(s_t)\right]. \quad (7.1)$$

Proof. We consider an algorithm that is no-regret with respect to the best-in-hindsight hypothesis in $\mathcal{H}'$, e.g., Hedge, and then account for the difference between the two best-in-hindsight hypotheses within sets $\mathcal{H}'$ and $\mathcal{H}$:

$$\begin{aligned}
\mathbb{E}[\textsc{Average-Regret}] &= \mathop{\mathbb{E}}_{\mathbf{s}}\left[\mathop{\mathrm{err}}_{\mathbf{s}}(h'_t) - \inf_{h\in\mathcal{H}}\mathop{\mathrm{err}}_{\mathbf{s}}(h)\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{s}}\left[\mathop{\mathrm{err}}_{\mathbf{s}}(h'_t) - \inf_{h'\in\mathcal{H}'}\mathop{\mathrm{err}}_{\mathbf{s}}(h')\right] + \mathop{\mathbb{E}}_{\mathbf{s}}\left[\inf_{h'\in\mathcal{H}'}\mathop{\mathrm{err}}_{\mathbf{s}}(h') - \inf_{h\in\mathcal{H}}\mathop{\mathrm{err}}_{\mathbf{s}}(h)\right] \\
&\leq O\left(\sqrt{\frac{\ln(|\mathcal{H}'|)}{T}}\right) + \mathop{\mathbb{E}}_{\mathbf{s}}\left[\sup_{h\in\mathcal{H}}\inf_{h'\mathcal{H}'}\left(\mathop{\mathrm{err}}_{\mathbf{s}}(h') - \mathop{\mathrm{err}}_{\mathbf{s}}(h)\right)\right] \\
&\leq O\left(\sqrt{\frac{\ln(|\mathcal{H}'|)}{T}}\right) + \frac{1}{T}\mathop{\mathbb{E}}_{\mathbf{s}}\left[\sup_{h\in\mathcal{H}}\inf_{h\in\mathcal{H}'}\sum_{t=1}^{T}\mathbb{1}(h'(s_t)\neq h(s_t))\right],
\end{aligned}$$

where the penultimate inequality holds by the guarantees of Hedge and re-ordering the terms. □

For the above lemma to be effective, we need to design a hypothesis class $\mathcal{H}'$ that is small (so the first term in (7.1) is small) yet representative of $\mathcal{H}$ (so that the second term in (7.1) is small). To capture this, we define a distance metric between two hypotheses h and h' by $d(h,h') = \Pr_{x\sim\mathcal{U}}[h(x)\neq h'(x)]$, where, to recall, $\mathcal{U}$ is the uniform distribution over the set $\mathcal{X}$. A hypothesis class $\mathcal{H}'$ is called a γ-*cover* of $\mathcal{H}$, if

$$\forall h\in\mathcal{H}, \exists h'\in\mathcal{H}', \text{such that } d(h,h')\leq\gamma.$$

The next lemma shows that a γ-cover of relatively small size exists for any class of hypotheses with small VC dimension.

Lemma 7.3.2. *Let $\mathcal{H}$ have VC dimension d. Then $\mathcal{H}$ has a γ-cover of size at most $(41/\gamma)^d$.*

Proof. We use the following ℓ_1 packing lemma.[1]

Lemma 4.11 of [11] *For any m and γ, let $G \subseteq \{0,1\}^m$ be such that for all $g, g' \in G$, $|\{i : g_i \neq g'_i\}| > \gamma m$. Then, $|G| \leq \left(\frac{41}{\gamma}\right)^d$, where d is the VC dimension of G.*

Suppose for contradiction that $\mathcal{H}$ has no γ-cover of size at most $(41/\gamma)^d$. Then, we can find $p > (41/\gamma)^d$ hypothesis $h_1, \ldots, h_p$ with $d(h_i, h_j) \geq \gamma + \delta$ for every $i \neq j$ (for some $\delta > 0$). Now consider m samples from $\mathcal{U}$. With probability approaching 1 as $m \to \infty$, for every $i \neq j$, h_i and h_j differ on more than a γ fraction of samples. Interpreting the labels of each hypothesis as a vector in $\{0,1\}^m$, this contradicts the above lemma. □

To effectively bound the second term of Equation (7.1), we need to show that any $h \in \mathcal{H}$ closely agrees with an $h' \in \mathcal{H}'$ on the instances $\mathbf{s} = (s_1, \ldots, s_T)$ that are generated by a smooth adversary. In this regard, a γ-cover only implies that any $h \in \mathcal{H}$ closely agrees with an $h' \in \mathcal{H}'$ in *expectation* over instances that are drawn from the *uniform distribution*. At a high level, a σ-smooth distribution resembles a uniform distribution: Since the density of a σ-smooth distribution cannot be overly skewed towards the disagreement region, any h and h' that usually agree over the uniform distribution also agree, slightly less usually, over a σ-smooth distribution. More formally, for any σ-smooth distribution $\mathcal{D}$,

$$\sup_{h \in \mathcal{H}} \inf_{h' \in \mathcal{H}} \Pr_{x \sim \mathcal{D}}[h(x) \neq h'(x)] \leq \sup_{h \in \mathcal{H}} \inf_{h' \in \mathcal{H}} \frac{\Pr_{x \sim \mathcal{U}}[h(x) \neq h'(x)]}{\sigma} \leq \frac{\gamma}{\sigma}. \tag{7.2}$$

What remains to show is that such an h and its corresponding h' closely agree, not just in expectation for a single sample, but also over the samples that are generated by the σ-smooth adversary. To simplify this, we capture the symmetric difference of hypotheses in $\mathcal{H}$ and their representatives in $\mathcal{H}'$ by a class of hypotheses $\mathcal{A}$, where

$$\mathcal{A} = \left\{ a \;\middle|\; \text{for some } h \in \mathcal{H} \text{ and } h' = \arg\min_{h' \in \mathcal{H}'} d(h, h'),\ a(x) = \mathbb{1}(h(x) \neq h'(x)) \right\}. \tag{7.3}$$

That is, $\mathcal{A}$ captures the disagreement regions between hypotheses of $\mathcal{H}$ and their nearest neighbors in $\mathcal{H}'$. Note that because $\mathcal{H}$ and $\mathcal{H}'$ both have VC dimension at most d, and $\mathcal{A}$ is a subset of their symmetric differences, Sauer's lemma shows that $\mathcal{A}$ has VC dimension $O(d)$.

We would like to show that the following generalization bound is small:

$$\mathop{\mathbb{E}}_{\mathbf{s}} \left[\sup_{a \in \mathcal{A}} \frac{1}{T} \sum_{t=1}^{T} a(x_t) - \mathop{\mathbb{E}}_{\mathbf{s}'} \left[\frac{1}{T} \sum_{t=1}^{T} a(x'_t) \right] \right], \tag{7.4}$$

where $s_t = (x_t, y_t)$ and $s'_t = (x'_t, y'_t)$ are generated by the adversary.

Indeed, for a non-adaptive σ-smooth adversary we prove a much stronger *uniform convergence bound*. That is for any class of hypotheses, beyond just $\mathcal{A}$, and regardless of the smoothness property, the values of all hypotheses on a sufficiently large sample set are close to their expectations.

[1] Note that the growth function of $\mathcal{H}$ can be used directly for a slightly worst dependence. We instead use an ℓ_1 packing bound that is obtained via chaining and directly yields a sharper upper bound on the cover size.

Lemma 7.3.3. *For any non-adaptive sequence of distributions $\mathcal{D}$ (that are not necessarily smooth) and any hypothesis class $\mathcal{G}$ with VC dimension d,*

$$\mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}}\left[\sup_{g\in\mathcal{G}}\frac{1}{T}\sum_{t=1}^{T}g(x_t)-\mathop{\mathbb{E}}_{\mathbf{s}'\sim\mathcal{D}}\left[\frac{1}{T}\sum_{t=1}^{T}g(x'_t)\right]\right]\leq O\left(\sqrt{\frac{d\ln(T/d)}{T}}\right),$$

where $s_t=(x_t,y_t)$ and $s'_t=(x'_t,y'_t)$.

Proof. This proof closely resembles the proof of the uniform convergence theorem for the setting where instances $s_1,\ldots,s_T$ are all generated from the same distribution $\mathcal{D}$. We adapt that proof and generalize it to scenarios where s_t is generated independently from distribution $\mathcal{D}_t$, where the choice of $\mathcal{D}_t$ is also independent of instances $s_{t'}$ for $t'\neq t$. In short, the observation that allows us to consider instances that are generated from different distributions is that during the process of *symmetrization* on the *ghost sample*, an instance s_t and its corresponding instance s'_t that may get swapped were generated by the same distribution. Therefore, their symmetrization induces an independent zero-mean random variable whose sum across all t is strongly concentrated by the Heoffding bound.

In more detail, when $s_t=(x_t,y_t)$ and $s'_t=(x'_t,y'_t)$ are independently generated from $\mathcal{D}_t$ and $\boldsymbol{\epsilon}$ is a vector of T rademacher random variables, we have,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}}\left[\sup_{g\in\mathcal{G}}\frac{1}{T}\sum_{t=1}^{T}g(x_t)-\mathop{\mathbb{E}}_{\mathbf{s}'\sim\mathcal{D}}\left[\frac{1}{T}\sum_{t=1}^{T}g(x'_t)\right]\right] &\leq \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{s}'\sim\mathcal{D}}\left[\sup_{g\in\mathcal{G}}\frac{1}{T}\sum_{t=1}^{T}\left(g(s_t)-g(s'_t)\right)\right] \\
&\qquad\qquad\text{(Jensen's ineqality)} \\
&= \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{s}'\sim\mathcal{D}}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}\left[\sup_{g\in\mathcal{G}}\frac{1}{T}\sum_{t=1}^{T}\epsilon_t\left(g(s_t)-g(s'_t)\right)\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{s}'\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}\left[\sup_{g\in\mathcal{G}[\mathbf{s}\cup\mathbf{s}']}\frac{1}{T}\sum_{t=1}^{T}\epsilon_t\left(g(s_t)-g(s'_t)\right)\,\middle|\,\mathbf{s},\mathbf{s}'\right]\right].
\end{aligned}$$

For fixed $\mathbf{s}$ and $\mathbf{s}'$ and any fixed $g\in\mathcal{G}[\mathbf{s}\cup\mathbf{s}']$, the random variables $\epsilon_t\left(g(s_t)-g(s'_t)\right)$ are independent across all t, with mean 0 and range $[-1,1]$. Therefore, for every $\rho>0$ and every g, Hoeffding's inequality implies that

$$\Pr_{\boldsymbol{\epsilon}}\left[\left|\frac{1}{T}\sum_{t=1}^{T}\epsilon_t\left(g(s_t)-g(s'_t)\right)\right|>\rho\,\middle|\,\mathbf{s},\mathbf{s}'\right]\leq 2\exp(-2T\rho^2).$$

Taking a union bound over all choices of $g\in\mathcal{G}[\mathbf{s}\cup\mathbf{s}']$ implies that, for every $\mathbf{s}$ and $\mathbf{s}'$ and every $\rho>0$,

$$\Pr_{\boldsymbol{\epsilon}}\left[\sup_{g\in\mathcal{G}[\mathbf{s}\cup\mathbf{s}']}\left|\frac{1}{T}\sum_{t=1}^{T}\epsilon_t\left(g(s_t)-g(s'_t)\right)\right|>\rho\,\middle|\,\mathbf{s},\mathbf{s}'\right]\leq 2\pi_{\mathcal{G}}(2T)\cdot\exp(-2T\rho^2),$$

where $\pi_{\mathcal{G}}$ denotes the growth function of $\mathcal{G}$ (see Section 7.2). Using Lemma A.4 of [246], this implies that

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}\left[\sup_{g\in\mathcal{G}[\mathbf{s}\cup\mathbf{s}']}\left|\frac{1}{T}\sum_{t=1}^{T}\epsilon_t\left(g(s_t)-g(s'_t)\right)\right|\,\middle|\,\mathbf{s},\mathbf{s}'\right]\leq\frac{4+\sqrt{\log(\pi_{\mathcal{G}}(2T))}}{\sqrt{2T}}.$$

Taking expectation over all $\mathbf{s}$ and $\mathbf{s}'$ and using the fact that $\pi_{\mathcal{G}}(m) \leq (em/d)^d$ for all m, we have

$$\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{s}'\sim\mathbf{P}}\left[\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}\left[\sup_{g\in\mathcal{G}[\mathbf{s}\cup\mathbf{s}']}\left|\frac{1}{T}\sum_{t=1}^{T}\epsilon_t\left(g(s_t)-g(s'_t)\right)\right| \;\middle|\; \mathbf{s},\mathbf{s}'\right]\right] \leq \frac{4+\sqrt{d\ln(2eT/d)}}{\sqrt{2T}}.$$

□

Let's now see how these different ingredients fit together to prove a regret bound against non-adaptive adversaries.

Theorem 7.3.4 (Non-adaptive adversaries)**.** *Let $\mathcal{H}$ be a hypothesis class of VC dimension d and $\mathcal{D}$ be an unknown non-adaptive sequence of σ-smooth distributions. Then, there is an algorithm with regret*

$$\mathbb{E}[\textsc{Average-Regret}] \leq O\left(\sqrt{\frac{d\ln\left(\frac{T}{\sigma}\right)}{T}}\right).$$

Proof. Define $\mathcal{A}$ as above. Note that using linearity of expectation and inequality (7.2),

$$\mathop{\mathbb{E}}_{\mathbf{s}\sim\boldsymbol{\mathcal{D}}}\left[\frac{1}{T}\sum_{t=1}^{T}a(s_t)\right] = \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s_t\sim\mathcal{D}_t}\left[a(s_t)\right] \leq \frac{\gamma}{\sigma}.$$

Let $\gamma = T^{-1/2}\sigma$. Then, lemmas 7.3.1, 7.3.2, and 7.3.3, together imply that

$$\mathbb{E}[\textsc{Average-Regret}] \leq O\left(\sqrt{\frac{d\ln(1/\gamma)}{T}}\right)+\frac{\gamma}{\sigma}+O\left(\sqrt{\frac{d\ln(T/d)}{T}}\right) \leq O\left(\sqrt{\frac{d\ln(T/\sigma)}{T}}\right).$$

□

7.4 Lower Bound for Non-adaptive Non-smooth Adversaries

In this section, we show that there is a non-adaptive (and non-smooth) adversary that enforces a regret of $\Omega(1)$ on any online algorithm. Therefore, the driving force behind our strong regret bound in Theorem 7.3.4 is indeed the smoothness of the adversary.

Theorem 7.4.1. *There exists a class of hypothesis $\mathcal{H}$ with VC dimension 1, and a non-adaptive and non-smooth adversary, such that for any online algorithm*

$$\mathbb{E}\left[\textsc{Average-Regret}\right] \in \Omega(1).$$

Proof. Consider a full binary tree of depth T, as shown in Figure 7.1, whose nodes correspond to multiples of powers of $\frac{1}{2}$. Let $\mathcal{H}$ be the class of linear thresholds in one dimension. We first show that each root-to-leaf path in this tree corresponds to one adversarial sequence and each randomized online algorithm corresponds to a distribution over labelings of the nodes of this tree. We then use the minimiax theorem to show that the worst non-adaptive adversary can enforce a

regret on the best randomized algorithm that is the same as the regret that the best deterministic algorithm gets against a distribution of adversarial sequences. Finally, we construct a distribution over sequences on which any deterministic algorithm incurs a large regret.

Let $\mathbf{y} \in \{+1, -1\}^T$ represent a root-to-leaf path in a binary tree of depth T, where $+1$ and -1 refer to taking the *left child* and *right child*, respectively. Let $\mathbf{Y}$ be the set of all such paths and $\Delta(\mathbf{Y})$ represent the set of all distributions over $\mathbf{Y}$. Let $\boldsymbol{\pi}$ be a labeled full binary tree, such that for any path $\mathbf{y}$ of length at most T, $\boldsymbol{\pi}(\mathbf{y})$ corresponds to the label of the node corresponding to the path $\mathbf{y}$. Let $\boldsymbol{\Pi}$ be the set of all labeled full binary trees and $\Delta(\boldsymbol{\Pi})$ the set of all distributions over them.

Next, we describe the learner and adversary's strategies on this tree.

1. Adversarial sequence $\{(x_t, y_t)\}_{t\in[T]}$: A path $\mathbf{y} \in \mathbf{Y}$ corresponds to a sequence in which at time t the adversary plays instance (x_t, y_t) where x_t corresponds to $\boldsymbol{\pi}(y_1, \ldots, y_{t-1})$.

2. A (randomized) online algorithm: An online algorithm can be denoted by a distribution over labeled binary trees. Since for any node there is a unique path from the root to it, a node x_t presented to the algorithm at time t corresponds to a unique history of the sequence, $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$. Therefore, an online algorithm's (randomize) choice of action at time t, h_t, is only a function of x_t. Conditioning on the value of $h_t(x_t)$, the choice of h_t does not affect the utility or the regret of the algorithm. Therefore, we can represent the decision of the algorithm at time t by a probability distribution over assigning label $+1$ or -1 to x_t. This induces a distribution $Q \in \Delta(\boldsymbol{\Pi})$ over labeled trees.

Note that for any $\mathbf{y} \in \mathbf{Y}$, there is a linear threshold in $\mathcal{H}$ that is consistent with labeled samples (x_t, y_t) for all $t \in [T]$. This is due to the fact that, for any $y_1, \ldots, y_{t-1}$, we always take a path that arrives at x_t which falls between the left-most node with $+1$ label and the right-most node with the -1 label. Therefore, $\inf_{\boldsymbol{\pi}} \operatorname{err}_{\mathbf{y}}(\boldsymbol{\pi}) = 0$. So, it is suffices to show that

$$\inf_{Q\in\Delta(\boldsymbol{\Pi})} \sup_{\mathbf{y}\in\mathbf{Y}} \mathop{\mathbb{E}}_{\boldsymbol{\pi}\sim Q} \left[\operatorname*{err}_{\mathbf{y}}(\boldsymbol{\pi})\right] = \Omega(T).$$

By the minimax theorem, we have

$$\inf_{Q\in\Delta(\boldsymbol{\Pi})} \sup_{\mathbf{y}\in\mathbf{Y}} \mathop{\mathbb{E}}_{\boldsymbol{\pi}\sim Q} \left[\operatorname*{err}_{\mathbf{y}}(\boldsymbol{\pi})\right] = \sup_{Y\in\Delta(\mathbf{Y})} \inf_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \mathop{\mathbb{E}}_{\mathbf{y}\sim Y} \left[\operatorname*{err}_{\mathbf{y}}(\boldsymbol{\pi})\right],$$

where the right hand side of the above equation is the loss of any deterministic algorithm against a distribution over adversarial sequences. Now consider $Y \in \Delta(\mathbf{Y})$ that is the uniform distribution over all sequences in $\mathbf{Y}$. In other words, at time t the adversary chooses $y_t = +1$ or $y_t = -1$ each with probability $\frac{1}{2}$. Then, any fixed labeling $\boldsymbol{\pi}$ incurs an error of $\frac{1}{2}$ in expectation. Therefore, $\mathbb{E}\left[\operatorname{err}_{\mathbf{y}}(\boldsymbol{\pi})\right] \geq T/2$ for any $\boldsymbol{\pi}$. This completes the proof.

□

7.5 Discussion and Open Problem

In this chapter, we applied the smoothed analysis framework of Spielman and Teng [252] to the problem of *regret-minimization in online learning,* and showed that fundamentally stronger

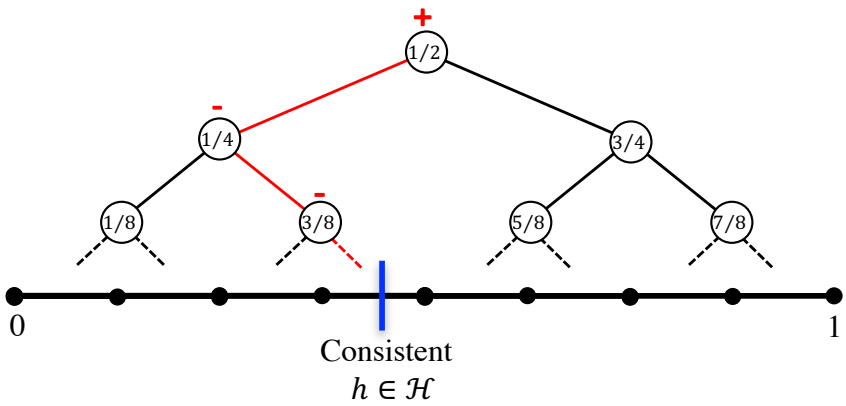


Figure 7.1: Path $(+1, -1, -1)$ is associated with the sequence $(\frac{1}{2}, +)$, $(\frac{1}{4}, -)$, and $(\frac{3}{8}, -)$.

regret guarantees are possible with smoothed adversaries than with worst-case adversaries. Our results demonstrate that, similar to offline (i.i.d) learning, the VC dimension of a hypothesis class characterizes the regret obtainable against smoothed adversaries. This is in stark contrast to the worse-case regret bounds that are characterized by the Littlestone dimension of a class of hypotheses, as the Littlestone dimension of even simple classes with constant VC dimension can be unbounded.

7.5.1 An Open Problem

Our main result in this chapter considers the case of non-adaptive smooth adversaries. Another adversarial model considered in online learning is the *adaptive* adversarial model. In this setting, the adversary makes a fresh choice at every time step, considering the entire history of the play until then. Interestingly, in the context of online non-smooth adversarial learning with full information, the minmax regret in the adaptive and non-adaptive settings are equal (See Section 7.4 and the work of Rakhlin and Sridharan [226]). A natural question is whether we can get improved regret bounds, as we did in the case of a non-adaptive smooth adversary, for adaptive smooth adversaries as well. We leave this general question as an open problem. But, we show some preliminary efforts in extending our results to the adaptive case.

Formally, an *adaptive* σ-smooth adversary is such that at every time step $t \in [T]$ the adversary adaptively chooses a σ-smooth distribution $\mathcal{D}_t$ based on the actions of the learner $h_1, \ldots, h_{t-1}$ and the realizations of the previous instances $s_1, \ldots, s_{t-1}$. We denote this random process by $\mathbf{s} \sim \boldsymbol{\mathcal{D}}$ and denote by $\mathrm{err}_{\boldsymbol{\mathcal{D}}}(h) = \mathbb{E}_{\mathbf{s}\sim\boldsymbol{\mathcal{D}}}[\mathrm{err}_{\mathbf{s}}(h)]$ the *true error* of hypothesis h on $\boldsymbol{\mathcal{D}}$.

As we observed in Lemma 7.3.3, any hypothesis class $\mathcal{G}$ with bounded VC dimension demonstrates uniform convergence on any non-adaptive sequence of distributions $\boldsymbol{\mathcal{D}}$, regardless of the smoothness of $\boldsymbol{\mathcal{D}}$ or other properties of $\mathcal{G}$. The next example demonstrates that this property fails to hold with adaptive adversaries.

Example 7.5.1. *Let $\mathcal{X} = [0, 1]$ and $\mathcal{G} = \{g_b(x) = \mathbb{1}(x \geq b) \mid \forall b \in [0, 1]\}$ be the set of one-dimensional thresholds. Let $\boldsymbol{\mathcal{D}}$ be an adaptive sequence of distributions, such that $\mathcal{D}_1$ is a uniform distribution over $[0, \frac{1}{4}] \cup [\frac{3}{4}, 1]$ and $\mathcal{D}_2 = \cdots = \mathcal{D}_T$ all are the uniform distribution on $[0, \frac{1}{4}]$ if $x_1 \in [0, 1/4]$, and otherwise, are the uniform distribution on $[\frac{3}{4}, 1]$. In this case, we do*

not achieve concentration for any value of T, as

$$\frac{1}{T}\sum_{t=1}^{T} g_{0.5}(x_t) = \begin{cases} 0 & \textit{w.p. } 1/2 \\ 1 & \textit{w.p. } 1/2 \end{cases} \qquad \textit{and} \qquad \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathscr{D}}\left[\frac{1}{T}\sum_{t=1}^{T} g_{0.5}(x_t)\right] = \frac{1}{2}.$$

An even more troubling aspect of the above example is that $\mathscr{D}$ is $\frac{1}{4}$-smooth and $\mathcal{G}$ has a VC dimension of 1. So, smoothness of an adaptive distribution is not sufficient for obtaining uniform convergence over $\mathcal{G}$, even if $\mathcal{G}$ has a small VC dimension.

At a high level, one of the challenging aspects of the above example is that $\{x \mid g_{0.5}(x) = 1\}$ is relatively large. This allows an adaptive adversary to create a smooth $\mathscr{D}$ where the supports of $\mathcal{D}_2, \ldots, \mathcal{D}_T$ are all within the set $\{x \mid g(x) = 1\}$ if some constant-probability event takes place in the first round, and otherwise choosing smooth $\mathcal{D}_2, \ldots, \mathcal{D}_T$ whose supports are within the set $\{x \mid g(x) = 0\}$. So, not all hope is lost in obtaining a no-regret algorithm for smooth adaptive adversaries. Note that the hypotheses that we really care about are those in $\mathcal{A}$, each of which is the symmetric difference of two similar hypotheses. That is, $\Pr_{\mathcal{U}}[a(x) = 1]$ is small for all $a \in \mathcal{A}$. So, it may still be possible to use this property to obtain a good generalization bound over all $a \in \mathcal{A}$ and prove Equation (7.4).

Part II

Learning from People

Chapter 8

Learning with Bounded Noise

8.1 Introduction

Designing noise tolerant and computationally efficient learning algorithms has been a long-standing question in learning theory. In absence of noise—when the data is realizable—such algorithms exist in a number of concepts classes, including halfspaces and constant-degree polynomials [88]. However, the problem becomes significantly harder in the presence of label noise. How much harder? It depends on the concept class and the noise model. In this chapter, we focus on learning d-dimensional halfspaces, which are one of the most popular classifiers studied in both the theory and practice of machine learning. Moreover, we focus on the *bounded noise* model (also known as *Massart noise*), which is a realistic and well-studied noise model from statistical learning theory [65, 204]. Our goal is to design algorithms that achieve error $\textsc{Opt} + \epsilon$ for an arbitrarily small ϵ— where $\textsc{Opt}$ is the error of the best halfspace—and run in time $\mathrm{poly}\left(\frac{1}{\epsilon}\right)$ and d.

The bounded noise model considers a setting where the label of each instance $\mathbf{x}$ is flipped independently with probability $\eta(\mathbf{x}) < 1/2$. The adversary has control over choosing a different, and unknown, noise rate $\eta(\mathbf{x}) \leq \eta$ for every instance $\mathbf{x}$ with the only constraint that $\eta(\mathbf{x}) \leq \eta$. From a statistical point of view, it is well known that one can obtain an improved sample complexity bound of $O(\frac{d}{\epsilon})$ under this noise model, compared to the worst-case joint distributions that requires $\Omega(\frac{d}{\epsilon^2})$ samples [65]. In computational learning theory, this noise model has also been studied under the name of *malicious misclassification noise* [231, 251]. However due to its highly assymmetric nature, until our recent works [24, 25], no computationally efficient learning algorithms were known in this model. In this chapter, we provide the first computationally efficient algorithm achieving arbitrarily small excess error for learning halfspaces in presence of bounded noise when the marginal distribution of the instance space is isotropic log-concave, e.g., Gaussian.

Theoretical Motivation for Bounded Noise The work on computationally efficient algorithms for learning halfspaces has mostly focused on two different extremes. On one hand, for the very stylized random classification noise model (RCN), where each example $\mathbf{x}$ is flipped independently with equal probability η, several works have provided computationally efficient

algorithms that can achieve arbitrarily small excess error in polynomial time [32, 54, 245], all of which crucially exploit the high amount of symmetry present in the RCN noise. At the other extreme, there has been significant work on much more difficult and adversarial noise models where the adversary deterministically flips labels of an adversarially selected η fraction of the instances. The best results here however, not only require additional distributional assumptions about the marginal over the instance space, but they only achieve much weaker multiplicative approximation guarantees of error $c\text{OPT}$ for a large constant c [23, 171, 181]. In this respect, bounded noise falls between these two extremes, where the noise is not fully adversarial but also far from being stylized.

Practical Motivation for Bounded Noise In addition to being of theoretical interest as a middle ground between the adversarial and random classification noise models, there is also a plausible narrative about why this noise may appear in practice, and especially in crowd-sourced data sets: Consider the PAC learning framework, where there is a true target halfspace $h^*(\mathbf{x}) = \text{sign}(\mathbf{w}^* \cdot \mathbf{x})$ that perfectly labels all instances $\mathbf{x}$. Consider an infinitely large crowd of labelers, $1 - \eta$ fraction of whom know the target function h^* and η fraction may make mistakes in arbitrary ways. Note that there may be easy instances, i.e., many of the imperfect labelers label them correctly; and more difficult instances, i.e., only the perfect labelers know their correct label. So, any instance $\mathbf{x}$ that is labeled by a randomly chosen person from this crowd receives an incorrect label with probability $\eta(\mathbf{x}) \leq \eta$, where the instance-dependent noise rate $\eta(\mathbf{x})$ captures the varying degree of difficulty of an instance. Indeed, the bounded noise model is a generalization of the noise models more commonly used in crowdsourcing, including the David-Skene model [96].

8.1.1 Our Results

In this work, we provide the first computationally efficient algorithm achieving arbitrarily small excess error for learning halfspaces. That is, we show that there exists a polynomial time algorithm that can learn a halfspace of error $\text{OPT} + \epsilon$ and run in $\text{poly}(d, \frac{1}{\epsilon})$ when the underlying distribution is an isotropic log-concave distribution in $\Re^d$ and the noise of each example $\eta(\mathbf{x}) \leq \frac{1}{2} - \beta$ for some $\beta = O(1)$.

A result of this form—that can get arbitrarily close to OPT—was only known for random classification noise. In random classification noise, the error of each classifier scales uniformly under the observed labels. This is indeed the underlying reason behind the working of many single-shot optimization mechanisms.[1] When bounded noise is considered, however, the observed error of classifiers under bounded noise could change drastically in a non-uniform fashion. This is due to the fact that the adversary has control over choosing a different noise rate $\eta(\mathbf{x}) \leq \eta$ for every example $\mathbf{x}$. As we show, these correlations significantly degrade the quality of the outcome of common single-shot optimization algorithms, such as the averaging algorithm [245] and hinge loss minimization.

[1]When distributions are far from being isotropic, some single shot mechanisms are accompanied by pre-processing steps that remove outliers, e.g. the work of Blum et al. [54]

In face of these challenges, we take an entirely different approach than previously considered for random classification noise. We define an adaptively chosen sequence of optimization tasks within smaller and smaller boundary regions of our the current guess for what the target classifier is. We show that when the optimization task at every round finds a halfspace with small, but constant, error within a given region, it directs us closer and closer to the optimal classifier. This allows us to achieve arbitrarily small excess error rate overall. We discuss this method and the nature of the optimization task in more detail in Section 8.1.2.

An appealing feature of our algorithm is that it is naturally adaptable to the active learning framework, which has been intensively studied in recent years [92, 147, 149]. In this case, our algorithm only requests labels for instances that fall within the optimization region in each iteration. We show that our algorithm achieves a label complexity of $\mathrm{poly}\left(d, \ln\left(\frac{1}{\epsilon}\right)\right)$, which is exponentially smaller than the number of unlabeled samples it requires. The following theorem informally states these results.

Theorems 8.3.1 (informal). *Consider a joint distribution $\tilde{\mathcal{D}}$ over $\mathcal{X} \times \mathcal{Y}$, such that the marginal distribution on $\mathcal{X}$ is an isotropic log-concave distribution in $\mathbb{R}^d$ and the labels satisfy bounded noise condition for a constant β with halfspace $\mathbf{w}^*$. There is an algorithm that takes $\mathrm{poly}\left(d, \frac{1}{\epsilon}, \ln\left(\frac{1}{\delta}\right)\right)$ unlabeled samples, $\mathrm{poly}\left(d, \ln\left(\frac{1}{\epsilon}\right), \ln\left(\frac{1}{\delta}\right)\right)$ labeled samples, runs in time $\mathrm{poly}\left(d, \frac{1}{\epsilon}, \ln\left(\frac{1}{\delta}\right)\right)$, and finds a halfspace $\mathbf{w}$ that with probability $1-\delta$, $\|\mathbf{w} - \mathbf{w}^*\|_2 \leq \epsilon$.*

We note that the guarantees of the above theorem, that $\|\mathbf{w} - \mathbf{w}^*\|_2 \leq \epsilon$, indeed implies that $\mathbf{w}^*$ has an excess error of $O(\epsilon)$ compared to $\mathbf{w}^*$. This is both due to the fact that the marginal distribution in the above theorem is an isotropic log-concave distribution and that excess error is bounded above by disagreement of $\mathbf{w}$ and $\mathbf{w}^*$ (See Lemma 8.2.1).

Given that our algorithm is an adaptively chosen sequence of optimization tasks, one might wonder what guarantee one-shot optimization of easy-to-optimize functions, such as hinge loss minimization and averaging, have. In Section 8.4, we show that a simple but powerful AVERAGE algorithm [245] cannot recover the true halfspace. In Section 8.5, we show a strong negative result: for every τ and β, there is a noisy distribution $\tilde{\mathcal{D}}$ over $\Re^d \times \{0, 1\}$ satisfying bounded noise with parameter β and an $\epsilon > 0$, such that τ-scaled hinge loss minimization returns a classifier with excess error $\Omega(\epsilon)$. Indeed, this result is not restricted to hinge loss minimization. As we show in our work [25], this can extend to a large class of functions that satisfy some properties, such as continuity and other mild conditions.

8.1.2 Our Techniques

Our algorithm follows a localization technique inspired by the work of Balcan et al. [35]. Our algorithm is initialized by a classifier $\mathbf{w}_0$ with a $0/1$ error that is at most an appropriate small constant more than the error of $\mathbf{w}^*$ with respect to the observed labels. This difference is known as the *excess error*. The algorithm then proceeds in rounds, aiming to cut down the excess error by half in each round. By the properties of bounded noise (Lemma 8.2.1) and the log-concave distribution (Lemma 8.2.2, Part 3), excess error of a classifier is a linear function of its angle to $\mathbf{w}^*$. Therefore, our algorithm aims to cut the angle by half at each round and eventually will output a $\mathbf{w}$ that is close to $\mathbf{w}^*$.

Consider $\mathbf{w}_{k-1}$ with angle $\leq \alpha_k$ to $\mathbf{w}^*$. It can be shown that for a band of width $\gamma_{k-1} = \Theta(\alpha_k)$ around the separator $\mathbf{w}_{k-1}$, $\mathbf{w}_{k-1}$ makes most of its error in this band. Therefore, improving the accuracy of $\mathbf{w}_{k-1}$ in the band significantly improves the accuracy of $\mathbf{w}_{k-1}$ overall. When considering vectors that are at angle $\leq \alpha_k$ to $\mathbf{w}_{k-1}$, it can be shown that any vector $\mathbf{w}_k$ that achieves *a small enough constant excess error with respect to the distribution in the band*, indeed, enjoys a much stronger guarantee of having *excess error that is half of* $\mathbf{w}_{k-1}$ *overall.* Therefore, if such a vector $\mathbf{w}_k$ can be found efficiently in the presence of bounded noise, a classifier of excess error ϵ can be learned in $O(\ln(\frac{1}{\epsilon}))$ steps. In order to make the above method work we need to achieve two goals: a) achieve a constant excess error while tolerating noise rate of $\eta(\mathbf{x}) \leq \frac{1}{2} - \frac{\beta}{2}$ and b) the hypothesis output should be a halfspace.

On one hand, efficient proper learning methods, such as surrogate loss minimization in the band, readily achieve goal (b). However, convex surrogate loss functions are only a good approximation of the 0/1 loss when the noise is small enough. Since the noise in the band can be as high as $\frac{1}{2} - \frac{\beta}{2}$, this directly restricts the noise rate of bounded noise that can be tolerated with such methods. Indeed, our first work in this space [24] demonstrated that when hinge-loss minimization is used in the band, such a method only works if the probability of flipping the label is as small as $\approx 10^{-6}$, i.e., when β is very close to 1. On the other hand, the polynomial regression approach of Kalai et al. [171] learns halfspaces to an arbitrary excess error of ϵ with runtime $\mathrm{poly}\big(d, \exp\big(\mathrm{poly}\big(\frac{1}{\epsilon}\big)\big)\big)$ when the marginal distribution is log-concave, requiring no additional assumption on noise. Since the distribution in the band is also log-concave, this method can achieve *an arbitrarily small constant* excess error in the band thereby achieving goal (a). However, this algorithm outputs the sign of a polynomial $p(\cdot)$ as a hypothesis, which is not necessarily a halfspace.

Instead, our algorithm takes a novel two-step approach to find $\mathbf{w}_k$ for *any amount of noise*. This is done by first finding a polynomial p_k that has a small constant excess error in the band. To obtain such a polynomial, we choose $\mathrm{poly}\left(d, \ln\left(\frac{\ln(1/\epsilon)}{\delta}\right)\right)$ labeled samples from the distribution in the band and use the algorithm by Kalai et al. [171] to find a polynomial with a small enough but, importantly, *a constant excess error*, e_{KKMS}, in the band. Note that at this point p_k already satisfies goal (a) but it does not satisfy goal (b) as it is not a halfspace. At a high level, since p_k has a small excess error with respect to $\mathbf{w}^*$ in the band, using a structural property of bounded noise that connects the excess error and disagreement of a classifier with respect to $\mathbf{w}^*$ (Lemma 8.2.1), we can show that p_k is also close in classification to $\mathbf{w}^*$. Therefore, it suffices to agnostically learn a halfspace $\mathbf{w}_k$ to a constant error for samples in the band that are labeled based on $\mathrm{sign}(p(\cdot))$. To achieve this, we use localized hinge loss minimization in the band over a set of samples that are labeled based on predictions of p_k to find $\mathbf{w}_k$. Therefore, $\mathbf{w}_k$ is close in classification to p_k in the band, which is in turn close to $\mathbf{w}^*$ in the band. As a result, $\mathbf{w}_k$ also has a small error in the band as desired.

8.1.3 Related Work

Learning linear classifiers under noise has been extensively studied in the past. One of the noise models considered in the past is the random classification noise (RCN) model [176]. Blum et al. [54] provided the first polynomial time algorithm capable of learning halfspaces in $\mathbb{R}^d$ to an

arbitrary accuracy ϵ under the RCN model. The algorithm works under any data distribution and runs in time polynomial in d, $\frac{1}{\epsilon}$ and $\frac{1}{1-2\eta}$, where $\eta < \frac{1}{2}$ is the probability of flipping a label under the RCN model. A simpler algorithm was later proposed by Dunagan and Vempala [114]. At the other extreme is the agnostic noise model where no assumption is made on the nature of the noise. In other words, the label of each example can be flipped in a completely adversarial fashion [177]. The goal is to output a hypothesis of error at most $\text{OPT} + \epsilon$, where OPT is the error of the best hypothesis in the class. Kalai et al. [171] designed an algorithm for learning halfspaces in this model under log-concave distributions. The algorithm relies on a structural result that under log-concave distributions, halfspaces are approximated in L_2 norm to ϵ accuracy, by a polynomial of degree $f(1/\epsilon)$. Here, $f(\cdot)$ is an exponentially growing function. Hence, by minimizing the absolute loss between the observed labels and a degree $f(1/\epsilon)$ polynomial, one can get arbitrarily close to the error of the best halfspace. Because the analysis does not rely on the existence of a good margin, the algorithm runs in time $d^{f(1/\epsilon)}$ and hence, is efficient only if ϵ is a constant. Shalev-Shwartz et al. [247] extended the work of Kalai et al. to design a learning algorithm for halfspaces which works for any distribution with a good *margin*. The run time however, still has a mild exponential dependence on $1/\epsilon$. In the agnostic model, algorithms with run time polynomial in d and $\frac{1}{\epsilon}$ are known if one is allowed to output a hypothesis of error multiplicatively worse than OPT. The simple *averaging algorithm* achieves a multiplicative error of $O(\sqrt{\ln \frac{1}{\text{OPT}}})$ [171]. This was improved to an error of $O(\text{OPT})$ using a different algorithm by Awasthi et al. [23]. Later, Daniely [89] showed how to get error of $(1+\mu)\text{OPT} + \epsilon$ in time inverse exponential in μ. The last two results mentioned above for the agnostic model hold for isotopic log-concave distributions. There are computational lower bounds suggesting that such multiplicative guarantees cannot be obtained under arbitrary distributions [90].

A family of interesting noise models lie between the RCN model and the agnostic noise model. The two most popular are the *bounded (a.k.a Massart)* noise model and the more general *Tsybakov noise* model [65, 259]. These models can be viewed as *semi-random* adversarial. For instance, in the bounded noise model, an adversary can decide, for each example $\mathbf{x}$, the probability $\eta(\mathbf{x}) \leq \frac{1}{2} - \frac{\beta}{2}$ of flipping the label of $\mathbf{x}$. The actual label of $\mathbf{x}$ is then generated by flipping a coin of the given bias $\eta(\mathbf{x})$. The computational study of bounded noise in the learning theory community dates back to early 90's with the work of [231, 251] who studied this model under the name *Malicious Misclassification Noise*. However, except for very simple cases, such as intervals on the line or other classes of constant VC-dimension, efficient algorithms in this model had remained unknown until recently. A variant of bounded noise, where the flipping probability for each point is either $\eta(\mathbf{x}) = 0$ or $\eta(\mathbf{x}) = \eta$ has been considered as an important open problem in learning theory with the hope that understanding the complexities involved in this type of noise could shed light on the problem of learning disjunctions in the presence of noise [50]. From the statistical point of view, it is also known that under this models, it is possible to get faster learning rates [65]. However, computationally efficient algorithms were not known until our works [24, 25].

Many other noise models are studied in the literature as well. The most popular among them is the linear noise model, where one assumes that the probability of a flipping the label of $\mathbf{x}$ is proportional to $|\mathbf{w}^* \cdot \mathbf{x}|$, where $\mathbf{w}^*$ is the optimal classifier. Because of the highly symmetric nature of the noise, efficient algorithms for halfspaces are known under this model [100]. The

recent work of Feige et al. [117] studies a noise model where one is allowed to perturb inputs and models the problem as a zero-sum game between a learner, minimizing the expected error, and an adversary, maximizing the expected error.

8.2 Preliminaries

We use $\mathcal{X}$ to denote the domain of the samples and $\mathcal{Y}$ to denote the label set. We work in a setting where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{+1, -1\}$. We define the *sign* function as $\mathrm{sign}(x) = 1$ if $x \geq 0$ and -1 otherwise. We consider a noise process denoted by $\mathcal{N}_\beta$ that corrupts the labels according to bounded noise with parameter β. In this model, a joint distribution over $(\mathcal{X}, \mathcal{Y})$ satisfies the bounded noise condition with parameter $\beta > 0$, if

$$|\Pr(y = +1|\mathbf{x}) - \Pr(y = -1|\mathbf{x})| \geq \beta, \, \forall \mathbf{x} \in \mathcal{X}.$$

In other words, bounded noise is equivalent to the setting where an adversary constructs the distribution by flipping the label of each point $\mathbf{x}$ from $\mathrm{sign}(\mathbf{w}^* \cdot \mathbf{x})$ to $-\mathrm{sign}(\mathbf{w}^* \cdot \mathbf{x})$ with a probability $\eta(\mathbf{x}) \leq \frac{1-\beta}{2}$. As is customary, we will use *Bayes optimal classifier* to refer to $\mathbf{w}^*$, the vector generating the uncorrupted measurements.

For any halfspace $\mathbf{w}$, we denote the resulting classifier $h_{\mathbf{w}}(\mathbf{x}) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$. For any classifier $h : \mathcal{X} \mapsto \mathcal{Y}$, we define the error with respect to distribution $\mathcal{P}$ as $\mathrm{err}_{\mathcal{P}}(h) = \Pr_{(\mathbf{x},y)\sim\mathcal{P}}[h(\mathbf{x}) \neq y]$. We define the *excess* error of h as $\mathrm{err}_{\mathcal{P}}(h) - \mathrm{err}_{\mathcal{P}}(h_{\mathbf{w}^*})$. We use OPT to denote the error of the Bayes classifier, i.e., $\mathrm{err}_{\mathcal{P}}(h_{\mathbf{w}^*})$. When the distribution is clear from the context, we use $\mathrm{err}(h_{\mathbf{w}^*})$ instead of $\mathrm{err}_{\mathcal{P}}(h_{\mathbf{w}^*})$.

Our goal in this chapter is to find a halfspace that has small excess error. As we show in the following lemma, this is directly related to finding a halfspace that has a small disagreement with $\mathbf{w}^*$.

Lemma 8.2.1 (Bousquet et al. [65])**.** *Given a classifier $h : \mathcal{X} \mapsto \{+1, -1\}$ and distribution $\mathcal{P}$ satisfying bounded noise condition with parameter β, let $\mathbf{w}^*$ be the Bayes optimal classifier. Then we have*

$$\beta \Pr_{(\mathbf{x},y)\sim\mathcal{P}}[h(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] \leq \operatorname*{err}_{\mathcal{P}}(h) - \operatorname*{err}_{\mathcal{P}}(h_{\mathbf{w}^*}) \leq \Pr_{(\mathbf{x},y)\sim\mathcal{P}}[h(x) \neq h_{\mathbf{w}^*}(\mathbf{x})]. \tag{8.1}$$

We frequently examine the region within a specified margin of a given halfspace. For distribution $\mathcal{P}$, halfspace $\mathbf{w}$, and margin γ, we denote by $\mathcal{P}_{\mathbf{w},\gamma}$ the conditional distribution over the set $S_{\mathbf{w},\gamma} = \{\mathbf{x} \mid |\mathbf{w} \cdot \mathbf{x}| \leq \gamma\}$. We define the *$\tau$-hinge loss* of a halfspace $\mathbf{w}$ over a labeled instance $(\mathbf{x}, y)$ as

$$\ell_\tau(\mathbf{w}, \mathbf{x}, y) = \max\left(0, 1 - \frac{y(\mathbf{w} \cdot \mathbf{x})}{\tau}\right).$$

When τ is clear from the context, we simply refer to the above quantity as the hinge loss. For a given set T of examples, we use $L_\tau(\mathbf{w}, T)$ to denote the empirical hinge loss over the set, i.e., $L_\tau(\mathbf{w}, T) = \frac{1}{|T|}\sum_{(\mathbf{x},y)\in T} \ell_\tau(\mathbf{w}, \mathbf{x}, y)$. For a classifier $\mathbf{w} \in \mathbb{R}^d$ and a value r, we use $B(\mathbf{w}, r)$ to denote the set $\{\mathbf{v} \in \mathbb{R}^d : \|\mathbf{w} - \mathbf{v}\|_2 \leq r\}$. Moreover, for two unit vectors $\mathbf{u}$ and $\mathbf{v}$, we use $\theta(\mathbf{u}, \mathbf{v}) = \arccos(\mathbf{u} \cdot \mathbf{v})$ to denote the angle between the two vectors.

In this chapter, we focus on distributions whose marginal over $\mathcal{X}$ is an *isotropic log-concave* distribution. A distribution in $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ with density function $f(\mathbf{x})$ is log-concave if $\ln(f(\mathbf{x}))$ is concave. In addition, the distribution is isotropic if it is centered at the origin, and its covariance matrix is the identity, i.e., $\mathbb{E}[x_i] = 0$, $\mathbb{E}[{x_i}^2] = 1$ for all i, and $\mathbb{E}[x_i x_j] = 0$ for all $i \neq j$. Below we state useful properties of such distributions. See [33, 197] for a proof of Lemma 8.2.2.

Lemma 8.2.2. *Let $\mathcal{P}$ be an isotropic log-concave distribution in $\mathbb{R}^d$. Then there exist absolute constants C_1, C_2 and C_3 such that*

1. *When $d = 1$, $\Pr_{x\sim\mathcal{P}}[x \geq \alpha] \leq \exp(-\alpha + 1)$.*

2. *All marginals of $\mathcal{P}$ are isotropic log-concave.*

3. *For any two unit vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^d$,*

$$C_1\theta(\mathbf{v}, \mathbf{u}) \leq \Pr_{\mathbf{x}\sim\mathcal{P}}[\mathrm{sign}(\mathbf{u}\cdot\mathbf{x}) \neq \mathrm{sign}(\mathbf{v}\cdot\mathbf{x})].$$

4. *For any unit vectors $\mathbf{w}$ and any γ,*

$$C_3\gamma \leq \Pr_{\mathbf{x}\sim\mathcal{P}}[|\mathbf{w}\cdot\mathbf{x}| \leq \gamma] \leq C_2\gamma.$$

5. *For any constant C_4, there exists a constant C_5 such that for two unit vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^d$ with $\|\mathbf{u} - \mathbf{v}\|_2 \leq r$ and $\theta(\mathbf{u}, \mathbf{v}) \leq \pi/2$, we have that*

$$\Pr_{\mathbf{x}\sim\mathcal{P}}[\mathrm{sign}(\mathbf{u}\cdot\mathbf{x}) \neq \mathrm{sign}(\mathbf{v}\cdot\mathbf{x}) \text{ and } |\mathbf{v}\cdot\mathbf{x}| \geq C_5 r] \leq C_4 r.$$

6. *For any constant C_6, there exists another constant C_7, such that for any unit vectors $\mathbf{v}$ and $\mathbf{u}$ in $\mathbb{R}^d$ such that $\|\mathbf{u} - \mathbf{v}\|_2 \leq r$ and any $\gamma \leq C_6$,*

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{P}_{\mathbf{u},\gamma}}\left[(\mathbf{v}\cdot\mathbf{x})^2\right] \leq C_7(r^2 + \gamma^2).$$

8.3 Bounded Noise Algorithm

In this section, we introduce efficient algorithms for recovering the true classifier in the presence of bounded noise for any constant β.

At a high level, our algorithm proceeds in $\log(\frac{1}{\epsilon})$ rounds and returns a halfspace $\mathbf{w}_k$ at round k whose disagreement with respect to $\mathbf{w}^*$ is halved at every step. We defer the discussion on how to find an appropriate initial classifier $\mathbf{w}_0$ to Section 8.3.2. By induction, consider $\mathbf{w}_{k-1}$

whose disagreement with $\mathbf{w}^*$ is at most $\Pr[\text{sign}(\mathbf{w}^* \cdot \mathbf{x}) \neq \text{sign}(\mathbf{w}_{k-1} \cdot \mathbf{x})] \leq \frac{\alpha_k}{\pi}$. First, we draw samples from the distribution of points that are at distance at most γ_{k-1} to $\mathbf{w}_{k-1}$. We call this region *the band* at round k and indicate it by $S_{\mathbf{w}_{k-1},\gamma_{k-1}}$. Next we apply the polynomial regression algorithm of [171] to get a polynomial $p(\cdot)$ of error a constant e_{KKMS} in the band. We draw additional samples from the band, label them based on $\text{sign}(p(\cdot))$, and minimize hinge loss with respect to these labels to get $\mathbf{w}_k$. We then show that $\mathbf{w}_k$ that is obtained using this procedure has disagreement at most $\frac{\alpha_{k+1}}{\pi}$ with the target classifier. We can then use $\mathbf{w}_k$ as the classifier for the next iteration. The detailed procedure is presented in Algorithm 8.1. The main result of this section is that Algorithm 8.1 efficiently learns halfspaces under log-concave distributions in the presence of bounded noise for any constant parameter β that is independent of the dimension. The small excess error implies arbitrarily small approximation rate to the optimal classifier $\mathbf{w}^*$ under bounded noise model.

Theorem 8.3.1. *Let the optimal Bayes classifier be a halfspace denoted by $\mathbf{w}^*$. Assume that the bounded noise condition holds for some constant $\beta \in (0, 1]$. For any $\epsilon > 0$, $\delta > 0$, there exist absolute constants e_0, C, C_1, C_2, c_1, c_2 such that Algorithm 8.1 with parameters $r_k = \frac{e_0}{C_1 2^k}$, $\gamma_k = Cr_k$, $\lambda = \frac{3C_1}{8CC_2}$, $e_{KKMS} = \beta(\lambda/(4c_1 + 4c_2 + 2))^4$, and $\tau_k = \lambda\ \gamma_{k-1}/(4c_1 + 4c_2 + 2)$ runs in polynomial time, proceeds in $s = O(\ln \frac{1}{\epsilon})$ rounds, where in round k it takes $n_k = \text{poly}(d, \exp(k), \ln(\frac{1}{\delta}))$ unlabeled samples and $m_k = \text{poly}(d, \ln(s/\delta))$ labels and with probability $1 - \delta$ returns a vector $\mathbf{w} \in \mathbb{R}^d$ such that $\|\mathbf{w} - \mathbf{w}^*\|_2 \leq \epsilon$.*

Algorithm 8.1: LEARNING HALFSPACES UNDER BOUNDED NOISE

1: Input: A sequence of values γ_k, τ_k and r_k for $k = 1, \ldots, \log(1/\epsilon)$, and e_{KKMS}.
2: Let $\mathbf{w}_0$ be the initial classifier as describe in Section 8.3.2.
3: **for** $k = 1, \ldots, \log(1/\epsilon) = s$ **do**
4: Take $\text{poly}(d, \ln(\frac{s}{\delta}))$ labeled samples from $\tilde{\mathcal{D}}_k$ and place them in the set T.
5: Perform polynomial regression [171] over T to find a polynomial p_k such that $\text{err}_{\tilde{\mathcal{D}}_k}(\text{sign}(p_k)) \leq \text{err}_{\tilde{\mathcal{D}}_k}(h_{\mathbf{w}^*}) + e_{\text{KKMS}}$.
6: Take $d(d + \ln(k/\delta))$ unlabeled samples from $\tilde{\mathcal{D}}_k$, label them according to $\text{sign}(p_k(\cdot))$. Call this set of labeled samples T'.
7: Find $\mathbf{v}_k \in B(\mathbf{w}_{k-1}, r_{k-1})$ that approximately minimizes the empirical hinge loss over T' using threshold τ_k, i.e., $L_{\tau_k}(\mathbf{v}_k, T') \leq \min_{w \in B(\mathbf{w}_{k-1}, r_{k-1})} L_{\tau_k}(\mathbf{w}, T') + \frac{\lambda}{12}$.
8: Let $\mathbf{w}_k = \frac{\mathbf{v}_k}{\|\mathbf{v}_k\|_2}$.
9: **end for**
10: **return** $\mathbf{w}_s$.

For the remainder of this chapter, we denote by $\tilde{\mathcal{D}}$ the noisy distribution and by $\mathcal{D}$ the distribution with labels corrected according to $\mathbf{w}^*$. Furthermore, we refer to $\tilde{\mathcal{D}}_{\mathbf{w}_{k-1},\gamma_{k-1}}$ and $\mathcal{D}_{\mathbf{w}_{k-1},\gamma_{k-1}}$, the noisy and clean distributions in the band, by $\tilde{\mathcal{D}}_k$ and $\mathcal{D}_k$, respectively.

8.3.1 Outline of the Proof and Related Lemmas

In this section, we provide an outline of the analysis of Algorithm 8.1 and the related lemmas. We defer the detailed proof of Theorem 8.3.1 to Section 8.3.3.

Consider a halfspace $\mathbf{w}_{k-1}$ at angle α_k to $\mathbf{w}^*$ and consider the band of width γ_{k-1} around $\mathbf{w}_{k-1}$. In a log-concave distribution, a $\Theta(\gamma_{k-1})$ fraction of the distribution falls in the band $S_{\mathbf{w}_{k-1},\gamma_{k-1}}$ (Property 4). Moreover, the probability that $\mathbf{w}_{k-1}$ makes a mistake outside of the band is a small constant fraction of α_k (Property 5). So, $\mathbf{w}_{k-1}$ makes most of its mistakes in the band $S_{\mathbf{w}_{k-1},\gamma_{k-1}}$. Therefore, if we can find a $\mathbf{w}_k$ that has a small (constant) error in the band and, similarly as in $\mathbf{w}_{k-1}$, is close to $\mathbf{w}^*$, then the overall error of $\mathbf{w}_k$ is a constant times better than that of $\mathbf{w}_{k-1}$. This is the underlying analysis of the margin-based technique [35]. It suffices to show that $\mathbf{w}_k$, indeed, has a small error rate (of a constant) in the band $S_{\mathbf{w}_{k-1},\gamma_{k-1}}$. That is, $\mathrm{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) \leq \lambda$ for the small constant λ. At each step of the algorithm, we first consider a polynomial $p(\cdot)$ obtained at Step 4 such that $\mathrm{err}(\mathrm{sign}(p(\cdot)) - \mathrm{err}(h_{\mathbf{w}^*}) \leq e_{\mathrm{KKMS}}$. Since the distribution in the band is also log-concave, we can use the polynomial regression algorithm of [171] to find such a polynomial.

Theorem 8.3.2 (Kalai et al. [171])**.** *Let $\mathcal{D}$ be a joint distribution over $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \in \{+1, -1\}$, such that the marginal over $\mathcal{X}$ is log-concave. Let* OPT *be the classification error of the best halfspace $\mathbf{w}^*$ with respect to $\mathcal{D}$. Then there exists an algorithm which, for any $\epsilon > 0$, outputs a polynomial $p(\cdot)$ such that $\mathrm{err}(\mathrm{sign}(p(\cdot)) \leq \mathrm{err}(h_{\mathbf{w}^*}) + \epsilon$. The running time and the number of samples needed by the algorithm is $d^{\exp(1/\epsilon^4)}$.*

Note, $\mathrm{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) \leq \Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[\mathrm{sign}(p_k(\mathbf{x})) \neq h_{\mathbf{w}^*}(\mathbf{x})] + \Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[h_{\mathbf{w}_k}(\mathbf{x}) \neq \mathrm{sign}(p_k(\mathbf{x}))]$. By the relation between the excess error and disagreement of a classifier under bounded noise (Lemma 8.2.1), polynomial p is e_{KKMS}/β close in classification to $\mathbf{w}^*$. Therefore, the first part of this inequality is at most e_{KKMS}/β. For the second part of this inequality we need to argue that $\mathbf{w}_k$ is close in classification to $p(\cdot)$ inside the band. Recall that at an intuitive level, we choose $\mathbf{w}_k$ so as to learn the labels of $p(\cdot)$. For this purpose, we draw samples from inside the band, label them based on $\mathrm{sign}(p(x))$, and then choose $\mathbf{w}_k$ that minimizes the hinge loss over these labels. Since this hinge loss is an upper bound on the disagreement of $p(\cdot)$ and $\mathbf{w}_k$, it suffices to show that it is small. We prove this in the following Lemma, where $\mathcal{D}'_k$ denotes the distribution $\mathcal{D}_k$ where the labels are predicted based on $\mathrm{sign}(p_k(\cdot))$.

The above discussion applies to the expected value of hinge loss. In the following lemma, we use VC dimension tools to show that for linear classifiers that are considered in Step 7 (the ones with angle α_k to $\mathbf{w}_k$), the empirical and expected hinge loss are close.

Lemma 8.3.3. *Let $\mathcal{D}'_k$ denote the distribution $\mathcal{D}_k$ where the labels are predicted based on $\mathrm{sign}(p_k(\cdot))$. There is $m_k = O(d(d + \ln(k/d)))$ such that for a randomly drawn set T' of m_k labeled samples from $\mathcal{D}'_k$, with probability $1 - \frac{\delta}{4(k+k^2)}$, for any $w \in B(\mathbf{w}_{k-1}, r_{k-1})$,*

$$\left| \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}[\ell_{\tau_k}(\mathbf{w}, \mathbf{x}, y)] - \ell_{\tau_k}(\mathbf{w}, T') \right| \leq \frac{\lambda}{12}.$$

Proof. The pseudo-dimension of the set of hinge loss values, i.e., $\{\ell_{\tau_k}(\mathbf{w}, \cdot) : \mathbf{w} \in \Re^d\}$ is known to be at most d. Next, we prove that for any halfspace $\mathbf{w} \in B(\mathbf{w}_{k-1}, r_{k-1})$ and for any point

$(\mathbf{x}, y) \sim \mathcal{D}'_k$, $\ell_{\tau_k}(\mathbf{w}, \mathbf{x}, y) \in O(\sqrt{d})$. We have,

$$\begin{aligned}
\ell_{\tau_k}(\mathbf{w}, \mathbf{x}, y) &\leq 1 + \frac{|\mathbf{w} \cdot \mathbf{x}|}{\tau_k} \\
&\leq 1 + \frac{|\mathbf{w}_{k-1} \cdot \mathbf{x}| + \|\mathbf{w} - \mathbf{w}_{k-1}\|_2 \|\mathbf{x}\|_2}{\tau_k} \\
&\leq 1 + \frac{\gamma_{k-1} + r_{k-1}\|\mathbf{x}\|_2}{\tau_k} \\
&\leq c(1 + \|\mathbf{x}\|_2).
\end{aligned}$$

By Lemma 8.2.2 part 1, for any $(\mathbf{x}, y) \in T'$, $\Pr_{(\mathbf{x},y)\sim\mathcal{D}'_k}[\|\mathbf{x}\|_2 > \alpha] \leq c\exp(-\alpha/\sqrt{d})$. Using union bound and setting $\alpha = \Theta(\sqrt{d}\ln(|T'|k^2/\delta))$ we have that with probability $1 - \frac{\delta}{8(k+k^2)}$, $\max_{\mathbf{x}\in T'} \|\mathbf{x}\|_2 \in O(\sqrt{d}\ln(|T'|k^2/\delta))$. Using standard pseudo-dimension rule we have that for $|T'| > \tilde{O}(d(d + \ln\frac{k}{\delta}))$, with probability $1 - \frac{\delta}{4(k+k^2)}$,

$$\left| \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}[\ell(\mathbf{w}, \mathbf{x}, y)] - \ell(\mathbf{w}, T') \right| \leq \frac{\lambda}{12}.$$

□

Next we show that the expected hinge of $\mathbf{w}^*$ with respect to $\mathcal{D}'$ is close to the expected hinge of $\mathbf{w}^*$ on the clean distribution $\mathcal{D}$ because p and $\mathbf{w}^*$ have small disagreement.

Lemma 8.3.4. *There exists an absolute constant c_2 such that*

$$\left| \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}\Big[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)\Big] - \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}_k}\Big[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)\Big] \right| \leq c_2 \frac{\gamma_{k-1}}{\tau_k} \sqrt{\mathop{\mathrm{err}}_{\mathcal{D}_k}(p_k)}.$$

Proof. Let N indicate the set of points $(\mathbf{x}, y)$ on which p_k and $h_{\mathbf{w}^*}$ disagree. We have,

$$\begin{aligned}
&\left| \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}\Big[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)\Big] - \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}_k}\Big[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)\Big] \right| \\
&\qquad \leq \left| \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}\Big[\mathbb{I}_{\mathbf{x}\in N}\big(\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y) - \ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, \mathrm{sign}(\mathbf{w}^* \cdot \mathbf{x}))\big)\Big] \right| \\
&\qquad \leq 2 \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}\left[\mathbb{I}_{\mathbf{x}\in N}\left(\frac{|\mathbf{w}^* \cdot \mathbf{x}|}{\tau_k}\right)\right] \\
&\qquad \leq \frac{2}{\tau_k}\sqrt{\Pr_{(\mathbf{x},y)\sim\mathcal{D}'_k}[\mathbf{x} \in N]} \times \sqrt{\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}[(\mathbf{w}^* \cdot \mathbf{x})^2]} \qquad \text{(By Cauchy Schwarz)} \\
&\qquad \leq \frac{2}{\tau_k}\sqrt{\mathop{\mathrm{err}}_{\mathcal{D}_k}(p_k)} \times \sqrt{\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}_k}[(\mathbf{w}^* \cdot \mathbf{x})^2]} \qquad \text{(By definition of } N) \\
&\qquad \leq \frac{2}{\tau_k}\sqrt{\mathop{\mathrm{err}}_{\mathcal{D}_k}(p_k)} \times \sqrt{C_7(r_{k-1}^2 + \gamma_{k-1}^2)} \qquad \text{(By Lemma 6)} \\
&\qquad \leq c_2 \frac{\gamma_{k-1}}{\tau_k}\sqrt{\mathop{\mathrm{err}}_{\mathcal{D}_k}(p_k)}.
\end{aligned}$$

□

Lemma 8.3.5. *There exists an absolute constant c_2 such that with probability $1 - \frac{\delta}{2(k+k^2)}$,*

$$\operatorname*{err}_{\mathcal{D}'_k}(h_{\mathbf{w}_k}) \leq 2 \operatorname*{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}_k}[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)] + 2c_2 \frac{\gamma_{k-1}}{\tau_k} \sqrt{\operatorname*{err}_{\mathcal{D}_k}(p_k)} + \frac{\lambda}{2}.$$

Proof. First, we note that the true 0/1 error of $\mathbf{w}_k$ on any distribution is at most its true hinge loss on that distribution. So, it suffices to bound the hinge loss of $\mathbf{w}_k$ on $\mathcal{D}'_k$. Moreover, $\mathbf{v}_k$ approximately minimizes the hinge loss on distribution $\mathcal{D}'_k$, so in particular, it performs better than $\mathbf{w}^*$ on $\mathcal{D}'_k$. On the other hand, Lemma 8.3.4 shows that the difference between hinge loss of $\mathbf{w}^*$ on $\mathcal{D}'_k$ and $\mathcal{D}_k$ is small. So, we complete the proof by using Lemma 8.3.6 and bounding the hinge of $\mathbf{w}^*$ on $\mathcal{D}_k$. The following equations show the process of derivation of this bound as we explained.

$$\begin{aligned}
&\operatorname*{err}_{\mathcal{D}'_k}(h_{\mathbf{w}_k}) \leq \operatorname*{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}'_k}[\ell_{\tau_k}(\mathbf{w}_k, \mathbf{x}, y)] && \text{(Since hinge loss larger than 0/1 loss)} \\
&\leq 2 \operatorname*{\mathbb{E}}_{(x,y)\sim\mathcal{D}'_k}[\ell_{\tau_k}(\mathbf{v}_k, \mathbf{x}, y)] && \text{(Since } \|\mathbf{v}_k\|_2 > 0.5\text{)} \\
&\leq 2L_{\tau_k}(\mathbf{v}_k, T') + 2(\frac{\lambda}{12}) && \text{(By Lemma 8.3.3)} \\
&\leq 2L_{\tau_k}(\mathbf{w}^*, T') + 4(\frac{\lambda}{12}) && (v_k \text{ was an approximate hinge loss minimizer}) \\
&\leq 2 \operatorname*{\mathbb{E}}_{(x,y)\sim\mathcal{D}'_k}[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)] + 6(\frac{\lambda}{12}) && \text{(By Lemma 8.3.3)} \\
&\leq 2 \operatorname*{\mathbb{E}}_{(x,y)\sim\mathcal{D}_k}[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)] + 2c_2 \frac{\gamma_{k-1}}{\tau_k} \sqrt{\operatorname*{err}_{\mathcal{D}_k}(p_k)} + \frac{\lambda}{2} && \text{(By Lemma 8.3.4)} \\
&\leq 2c_1 \frac{\tau_k}{\gamma_{k-1}} + 2c_2 \frac{\gamma_{k-1}}{\tau_k} \sqrt{\operatorname*{err}_{\mathcal{D}_k}(p_k)} + \frac{\lambda}{2}. && \text{(By Lemma 8.3.6)}
\end{aligned}$$

□

Finally, we show that hinge loss of $\mathbf{w}^*$ on the clean distribution can be upper bounded by the parameters of the algorithm. Together with the result of Lemma 8.3.5 this shows that $\operatorname{err}_{\mathcal{D}'_k}(\mathbf{w}_k) \leq \lambda$ as desired.

Lemma 8.3.6. *There exists an absolute constant c_1 such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_k}[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)] \leq c_1 \frac{\tau_k}{\gamma_{k-1}}$.*

Proof. Notice that $\mathbf{w}^*$ never makes a mistake on distribution $\mathcal{D}_k$, so the hinge loss of $\mathbf{w}^*$ on $\mathcal{D}_k$ is entirely attributed to the points of $\mathcal{D}_k$ that are within distance τ_k from $\mathbf{w}^*$. We have,

$$\begin{aligned}
\operatorname*{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}_k}[\ell_{\tau_k}(\mathbf{w}^*, \mathbf{x}, y)] &\leq \Pr_{(x,y)\sim\mathcal{D}_k}[|\mathbf{w}^* \cdot \mathbf{x}| < \tau_k] \\
&= \frac{\Pr_{(x,y)\sim\mathcal{D}}[|\mathbf{w}^* \cdot \mathbf{x}| < \tau_k]}{\Pr_{(x,y)\sim\mathcal{D}}[|\mathbf{w}_{k-1} \cdot \mathbf{x}| \leq \gamma_{k-1}]} \\
&\leq \frac{C_2\tau_k}{C_3\gamma_{k-1}} && \text{(By Part 4 of Lemma 8.2.2)} \\
&\leq c_1 \frac{\tau_k}{\gamma_{k-1}}.
\end{aligned}$$

□

8.3.2 Initializing $\mathbf{w}_0$

We need to find $\mathbf{w}_0$ such that $\text{err}_{\mathcal{D}}(h_{\mathbf{w}_0}) \leq \frac{\pi}{2C_1}$. To find such $\mathbf{w}_0$, we first take a labeled sample of size $m_0 = \text{poly}(d, \ln(\ln(1/\epsilon)/\delta))$ from $\tilde{\mathcal{D}}$ and run the polynomial regression algorithm of Kalai et al. [171] on this set to find a polynomial $p_0(\cdot)$ with excess error $e'_{\text{KKMS}} = \beta\left(\frac{\pi}{4(1+C'_1+C'_2)C_1}\right)^4$, where we defer the choice of C'_1 and C'_2 to later in the proof.

Then we take T' a sample of size $\text{poly}(d, \ln(1/\delta))$ from $\mathcal{D}$ and label it based on $\text{sign}(p_0(\cdot))$. We find $\mathbf{w}_0 = \frac{\mathbf{v}_0}{\|\mathbf{v}_0\|_2}$ that approximately minimizes the empirical hinge loss over this sample. We have

$$\underset{\mathcal{D}}{\text{err}}(h_{\mathbf{w}_0}) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[p_0(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] + \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[p_0(\mathbf{x}) \neq h_{\mathbf{w}_0}(\mathbf{x})].$$

For the second part of this equation, we follow a similar proof to the analysis of Lemma 8.3.5. Note that for any unit length $\mathbf{w}$ and $\mathbf{x}$ drawn from an isotropic log-concave distribution $\mathbf{w} \cdot \mathbf{x}$ is also an isotropic log-concave distribution, so $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(\mathbf{w} \cdot \mathbf{x})^2] = 1$. Using this in Lemmas 8.3.4 and 8.3.6, using a constant generalization error κ, and $\tau = \text{err}_{\mathcal{D}'}(h_{\mathbf{w}^*})^{1/4}$, we have

$$\begin{aligned}
\underset{\mathcal{D}'}{\text{err}}(h_{\mathbf{w}_0}) &\leq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}'}[\ell_\tau(\mathbf{w}_0, \mathbf{x}, y)] && \text{(hinge loss is greater than 0/1 loss)} \\
&\leq 2\,\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}'}[\ell_\tau(\mathbf{v}_0, \mathbf{x}, y)] && \text{(Since } \|\mathbf{v}_0\|_2 > 0.5\text{)} \\
&\leq 2L_\tau(\mathbf{v}_0, T') + 2\left(\frac{\kappa}{6}\right) && \text{(By Lemma 8.3.3)} \\
&\leq 2L_\tau(\mathbf{w}^*, T') + 4\left(\frac{\kappa}{6}\right) && \text{(}\mathbf{v}_0\text{ is an approximate hinge loss minimizer)} \\
&\leq 2\,\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}'}[\ell_\tau(\mathbf{w}^*, \mathbf{x}, y)] + \kappa && \text{(By Lemma 8.3.3)} \\
&\leq 2\,\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell_\tau(\mathbf{w}^*, \mathbf{x}, y)] + \frac{2C'_2}{\tau}\sqrt{\underset{\mathcal{D}}{\text{err}}(p_0)} + \kappa && \text{(By Lemma 8.3.4)} \\
&\leq 2C'_1\tau + \frac{2C'_2}{\tau}\sqrt{\underset{\mathcal{D}}{\text{err}}(p_0)} + \kappa && \text{(By Lemma 8.3.6)} \\
&\leq (2C'_1 + 2C'_2)\left(\underset{\mathcal{D}}{\text{err}}(p_0)\right)^{1/4} + \kappa.
\end{aligned}$$

For $\kappa = e'_{\text{KKMS}}$ we have

$$\begin{aligned}
\underset{\mathcal{D}}{\text{err}}(h_{\mathbf{w}_0}) &\leq \underset{\mathcal{D}}{\text{err}}(p_0) + \underset{\mathcal{D}'}{\text{err}}(h_{\mathbf{w}_0}) \\
&\leq (2 + 2C'_1 + 2C'_2)\left(\underset{\mathcal{D}}{\text{err}}(p_0)\right)^{1/4} \\
&\leq (2 + 2C'_1 + 2C'_2)\left(\frac{1}{\beta}e'_{\text{KKMS}}\right)^{1/4} && \text{(By Lemma 8.2.1)} \\
&\leq \frac{\pi}{2C_1}.
\end{aligned}$$

8.3.3 Putting Everything Together

Here, we formally prove Theorem 8.3.1. Recall that we use the following parameters in Algorithm 8.1: $r_k = \frac{e_0}{C_1 2^k}$, $\gamma_k = Cr_k$, where we defer the choice of C to later in the proof, $\lambda = \frac{3C_1}{8CC_2}$, $e_{\text{KKMS}} = \beta(\lambda/(4c_1 + 4c_2 + 2))^4$, and $\tau_k = \lambda\gamma_{k-1}/(4c_1 + 4c_2 + 2)$. Note, that by Lemma 8.2.1, for any classifier h,

$$\operatorname*{err}_{\tilde{\mathcal{D}}}(h) - \operatorname*{err}_{\tilde{\mathcal{D}}}(h_{\mathbf{w}^*}) \leq \Pr_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}}[h(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] = \operatorname*{err}_{\mathcal{D}}(h).$$

In this proof, we show that Algorithm 8.1 returns $\mathbf{w}_s$ such that $\operatorname{err}_{\mathcal{D}}(h_{\mathbf{w}_s}) \leq \epsilon$, and in turn, the excess error of $h_{\mathbf{w}_s}$ is also at most ϵ.

For the initialization of $\mathbf{w}_0$, using the analysis of Section 8.3.2, we have that $e_0 \leq \frac{\pi}{2C_1}$. Using Lemma 8.2.2, part 3, this shows that $\theta(\mathbf{w}_0, \mathbf{w}^*) \leq \frac{\pi}{2}$. Note that this gives us $\theta(\mathbf{w}, \mathbf{w}^*) \leq \frac{\pi}{2}$ in the rest of the algorithm, which is a pre-requisite for Lemma 8.2.2 part 5 and other lemmas.

We use induction to show that at the k^{th} step of the algorithm $\theta(\mathbf{w}_k, \mathbf{w}^*) \leq \frac{e_0}{C_1 2^k}$. Using Lemma 8.2.2, part 3, it suffices to show that $\operatorname{err}_{\mathcal{D}}(h_{\mathbf{w}_k}) \leq e_0/2^k$. Assume by the induction hypothesis that at round $k-1$, $\operatorname{err}_{\mathcal{D}}(h_{\mathbf{w}_{k-1}}) \leq e_0/2^{k-1}$. We will show that $\mathbf{w}_k$, which is chosen by the algorithm at round k, also has the property that $\operatorname{err}_{\mathcal{D}}(h_{\mathbf{w}_k}) \leq e_0/2^k$.

Let $S_k = \{\mathbf{x} \mid |\mathbf{w}_{k-1} \cdot \mathbf{x}| \leq \gamma_{k-1}\}$ indicate the band at round k. We divide the error of $\mathbf{w}_k$ to two parts, error outside the band and error inside the band. That is,

$$\operatorname*{err}_{\mathcal{D}}(h_{\mathbf{w}_k}) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbf{x} \notin S_k \text{ and } h_{\mathbf{w}_k}(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] + \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbf{x} \in S_k \text{ and } h_{\mathbf{w}_k}(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})]. \tag{8.2}$$

By Part 3 of Lemma 8.2.2, $\theta(\mathbf{w}_{k-1}, \mathbf{w}^*) \leq r_{k-1}$. So, for the first part of the above inequality, which is the error of $\mathbf{w}_k$ outside the band, we have that

$$\begin{aligned}
&\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbf{x} \notin S_k \text{ and } h_{\mathbf{w}_k}(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] \\
&\quad\leq \Pr_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbf{x} \notin S_k \text{ and } h_{\mathbf{w}_k}(\mathbf{x}) \neq h_{\mathbf{w}_{k-1}}(\mathbf{x})\right] + \Pr_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbf{x} \notin S_k \text{ and } h_{\mathbf{w}_{k-1}}(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})\right] \\
&\quad\leq 2\frac{C_1 r_{k-1}}{16} \leq \frac{e_0}{4 \times 2^k},
\end{aligned} \tag{8.3}$$

where the penultimate inequality follows from the fact that by the choice of $\mathbf{w}_k \in B(\mathbf{w}_{k-1}, r_{k-1})$ and the induction hypothesis, respectively, $\theta(\mathbf{w}_{k-1}, \mathbf{w}_k) < r_{k-1}$ and $\theta(\mathbf{w}_{k-1}, \mathbf{w}^*) < r_{k-1}$; By choosing large enough constant C in $\gamma_{k-1} = Cr_{k-1}$, using Part 5 of Lemma 8.2.2, the probability of disagreement outside of the band is $C_1 r_{k-1}/16$.

For the second part of Equation 8.2 we have that

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbf{x} \in S_k \text{ and } h_{\mathbf{w}_k}(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] = \operatorname*{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbf{x} \in S_k], \tag{8.4}$$

and

$$\operatorname*{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) \Pr_{(\mathbf{x},y)\sim D}[\mathbf{x} \in S_k] \leq \operatorname*{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) C_2 \gamma_{k-1} \leq \operatorname*{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) \frac{2C_2 C e_0}{C_1 2^k}, \tag{8.5}$$

where the penultimate inequality is based on Part 4 of Lemma 8.2.2. Therefore, by replacing Equations 8.3 and 8.5 with Equation 8.2, we see that in order to have $\mathrm{err}_{\mathcal{D}}(h_{\mathbf{w}_k}) < \frac{e_0}{2^k}$, it suffices to show that $\mathrm{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) \leq \frac{3C_1}{8CC_2} = \lambda$. The rest of the analysis is contributed to proving this bound. We have

$$\begin{aligned}\operatorname*{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) &= \Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[h_{\mathbf{w}_k}(\mathbf{x}) \neq h_{\mathbf{w}^*}(\mathbf{x})] \\ &\leq \Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[\mathrm{sign}(p_k(\mathbf{x})) \neq h_{\mathbf{w}^*}(\mathbf{x})] + \Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[h_{\mathbf{w}_k}(\mathbf{x}) \neq \mathrm{sign}(p_k(\mathbf{x}))].\end{aligned}$$

For the first part, using the assumption in Lemma 8.2.1, we have that

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[\mathrm{sign}(p_k(\mathbf{x})) \neq h_{\mathbf{w}^*}(\mathbf{x})] \leq \frac{1}{\beta}\left(\operatorname*{err}_{\tilde{\mathcal{D}}_k}(\mathrm{sign}(p_k)) - \operatorname*{err}_{\tilde{D}_k}(h_{\mathbf{w}^*})\right) \leq \frac{e_{\text{KKMS}}}{\beta}. \tag{8.6}$$

For the second part, using Lemma 8.3.5, we have

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}_k}[h_{w_k}(\mathbf{x}) \neq \mathrm{sign}(p_k(\mathbf{x}))] = \operatorname*{err}_{\mathcal{D}'_k}(h_{\mathbf{w}_k}) \leq 2c_1\frac{\tau_k}{\gamma_{k-1}} + 2c_2\frac{\gamma_{k-1}}{\tau_k}\sqrt{\frac{e_{\text{KKMS}}}{\beta}} + \frac{\lambda}{2}.$$

Therefore, by the choice of parameter $\tau_k = \lambda\gamma_{k-1}/(4c_1 + 4c_2 + 2) = \gamma_{k-1}(e_{\text{KKMS}}/\beta)^{1/4}$, we have

$$\begin{aligned}\operatorname*{err}_{\mathcal{D}_k}(h_{\mathbf{w}_k}) &\leq \frac{e_{\text{KKMS}}}{\beta} + 2c_1\frac{\tau_k}{\gamma_{k-1}} + 2c_2\frac{\gamma_{k-1}}{\tau_k}\sqrt{\frac{e_{\text{KKMS}}}{\beta}} + \frac{\lambda}{2} \\ &\leq \frac{e_{\text{KKMS}}}{\beta} + 2c_1\left(\frac{e_{\text{KKMS}}}{\beta}\right)^{1/4} + 2c_2\left(\frac{e_{\text{KKMS}}}{\beta}\right)^{1/4} + \frac{\lambda}{2} \\ &\leq (2c_1 + 2c_2 + 1)\left(\frac{e_{\text{KKMS}}}{\beta}\right)^{1/4} + \frac{\lambda}{2} \leq \frac{\lambda}{2} + \frac{\lambda}{2} \leq \lambda.\end{aligned}$$

Sample Complexity and Run-time To get error of e_{KKMS} with probability $1 - \frac{s}{\delta}$ at every round, we need a labeled set of size $\mathrm{poly}(d, \ln\frac{s}{\delta})$. The sample set T' is labeled based on p_k, so it does not contribute to the label complexity. So, at each round, we need $m_k = \mathrm{poly}(d, \ln(\frac{\ln(1/\epsilon)}{\delta}))$ labels. At each round, to get $\mathrm{poly}(d, \ln(\frac{\ln(1/\epsilon)}{\delta}))$ labels for the polynomial regression algorithm in the band of S_k we need $O(2^k m_k)$ samples from $\tilde{\mathcal{D}}$. To get $d(d + \ln(k/\delta))$ unlabeled samples in the band for Step 6, we need $O(2^k(d(d+\ln(k/\delta))) = \mathrm{poly}(d, \exp(k), \ln(\frac{1}{\delta}))$ unlabeled samples. So, overall, we need $n_k = \mathrm{poly}(d, \exp(k), \ln(\frac{1}{\delta}))$ unlabeled samples at each round. The running time is dominated by the polynomial regression algorithm which takes time $d^{\exp(\frac{1}{\beta^4})}$. However, since β is a constant, this is a polynomial in d.

8.4 AVERAGE Does Not Work

Our algorithm described in the previous section uses polynomial regression and hinge loss minimization in the band as an efficient proxy for minimizing the $0/1$ loss. The AVERAGE

algorithm introduced by Servedio [245] is another computationally efficient algorithm that has provable noise tolerance guarantees under certain noise models and distributions. For example, it achieves arbitrarily small excess error in the presence of random classification noise and monotonic noise when the distribution is uniform over the unit sphere. Furthermore, even in presence of a small amount of malicious noise and less symmetric distributions, AVERAGE has been used to obtain a weak learner, which can then be boosted to achieve a non-trivial noise tolerance [181]. Therefore it is natural to ask, *whether the noise tolerance that* AVERAGE *exhibits could be extended to the case of bounded noise under the uniform distribution?* We answer this question in the negative. We show that the lack of symmetry in bounded noise presents a significant barrier for the one-shot application of AVERAGE, even when the marginal distribution is completely symmetric. Additionally, we also discuss obstacles in incorporating AVERAGE as a weak learner—in place of polynomial regression and hinge loss minimization—within the margin-based framework.

In a nutshell, AVERAGE takes m sample points and their respective labels, $W = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^m, y^m)\}$, and returns $\frac{1}{m}\sum_{i=1}^{m} \mathbf{x}^i y^i$. Our main result in this section shows that for a wide range of distributions that are very symmetric in nature, including the Gaussian and the uniform distribution, there is an instance of bounded noise under which AVERAGE cannot achieve an arbitrarily small excess error.

Theorem 8.4.1. *For any continuous distribution $\mathcal{D}$ with a p.d.f. that is a function of the distance from the origin only, there is a noisy distribution $\tilde{\mathcal{D}}$ over $\mathcal{X} \times \{0, 1\}$ that satisfies bounded noise condition for some parameter $\beta > 0$ and* AVERAGE *returns a classifier with excess error $\Omega\left(\frac{\beta(1-\beta)}{1+\beta}\right)$.*

Proof. Let $\mathbf{w}^* = (1, 0, \ldots, 0)$ be the target halfspace. Let the noise distribution be such that for all $\mathbf{x}$, if $x_1 x_2 < 0$ then we flip the label of $\mathbf{x}$ with probability $\frac{1-\beta}{2}$, otherwise we keep the label. Clearly, this satisfies bounded noise with parameter β. Let $\mathbf{w}$ be the expected vector returned by AVERAGE. We first show that $\mathbf{w}$ is far from $\mathbf{w}^*$ in angle. Then, using Lemma 8.2.1 we show that $\mathbf{w}$ has large excess error.

First we examine the expected component of $\mathbf{w}$ that is parallel to $\mathbf{w}^*$, i.e., $\mathbf{w} \cdot \mathbf{w}^* = w_1$. For ease of exposition, we divide our analysis to two cases, one for regions with no noise (first and third quadrants) and second for regions with noise (second and fourth quadrants). Let $\mathcal{E}$ be the event that $x_1 x_2 > 0$. By symmetry, it is easy to see that $\Pr[\mathcal{E}] = 1/2$. Then

$$\mathbb{E}[\mathbf{w} \cdot \mathbf{w}^*] = \Pr(\mathcal{E})\ \mathbb{E}[\mathbf{w} \cdot \mathbf{w}^* | \mathcal{E}] + \Pr(\bar{\mathcal{E}})\ \mathbb{E}[\mathbf{w} \cdot \mathbf{w}^* | \bar{\mathcal{E}}]$$

For the first term, for $\mathbf{x} \in \mathcal{E}$ the label has not changed. So, $\mathbb{E}[\mathbf{w} \cdot \mathbf{w}^* | \mathcal{E}] = \mathbb{E}[|x_1|\ | \mathcal{E}] = \int_0^1 z f(z)$. For the second term, the label of each point stays the same with probability $\frac{1+\beta}{2}$ and is flipped with probability $\frac{1-\beta}{2}$. Hence, $\mathbb{E}[\mathbf{w} \cdot \mathbf{w}^* | \mathcal{E}] = \beta\ \mathbb{E}[|x_1|\ | \mathcal{E}] = \beta \int_0^1 z f(z)$. Therefore, the expected parallel component of $\mathbf{w}$ is $\mathbb{E}[\mathbf{w} \cdot \mathbf{w}^*] = \frac{1+\beta}{2} \int_0^1 z f(z)$

Next, we examine w_2, the orthogonal component of $\mathbf{w}$ on the second coordinate. Similar to the previous case for the clean regions $\mathbb{E}[w_2 | \mathcal{E}] = \mathbb{E}[|x_2|\ | \mathcal{E}] = \int_0^1 z f(z)$. Next, for the second and forth quadrants, which are noisy, we have

$$\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}}[x_2 y | x_1 x_2 < 0] = (\frac{1+\beta}{2}) \int_{-1}^{0} z \frac{f(z)}{2} + (\frac{1-\beta}{2}) \int_{-1}^{0} (-z) \frac{f(z)}{2} \qquad \text{(Fourth quadrant)}$$

$$
\begin{aligned}
&+ (\frac{1+\beta}{2}) \int_0^1 (-z) \frac{f(z)}{2} + (\frac{1-\beta}{2}) \int_0^1 z \frac{f(z)}{2} && \text{(Second quadrant)} \\
&= -(\frac{1+\beta}{2}) \int_0^1 z \frac{f(z)}{2} + (\frac{1-\beta}{2}) \int_0^1 z \frac{f(z)}{2} \\
&- (\frac{1+\beta}{2}) \int_0^1 z \frac{f(z)}{2} + (\frac{1-\beta}{2}) \int_0^1 z \frac{f(z)}{2} && \text{(By symmetry)} \\
&= -\beta \int_0^1 z f(z).
\end{aligned}
$$

So, $w_2 = \left(\frac{1-\beta}{2}\right) \int_0^1 z f(z)$. Therefore $\theta(\mathbf{w}, \mathbf{w}^*) = \arctan(\frac{1-\beta}{1+\beta}) \geq \frac{1-\beta}{(1+\beta)}$. By Lemma 8.2.1, we have $\mathrm{err}_{\tilde{\mathcal{D}}}(\mathbf{w}) - \mathrm{err}_{\tilde{\mathcal{D}}}(\mathbf{w}^*) \geq \beta \frac{\theta(\mathbf{w}, \mathbf{w}^*)}{\pi} \geq \beta \frac{1-\beta}{\pi(1+\beta)}$. □

Our algorithms rely on using polynomial regression and hinge-loss minimization in the band at every round to efficiently find a halfspace $\mathbf{w}_k$ that is a weak learner for $\mathcal{D}_k$, i.e., $\mathrm{err}_{\mathcal{D}_k}(\mathbf{w}_k)$ is at most a small constant. Motivated by this more lenient goal of finding a weak learner, one might ask whether AVERAGE, as an efficient algorithm for finding low error halfspaces, can be incorporated with the margin-based technique in the same way as hinge loss minimization? We argue that the margin-based technique is inherently incompatible with AVERAGE.

The Margin-based technique maintains two key properties at every step: First, the angle between $\mathbf{w}_k$ and $\mathbf{w}_{k-1}$ and the angle between $\mathbf{w}_{k-1}$and $\mathbf{w}^*$ are small, and as a result $\theta(\mathbf{w}^*, \mathbf{w}_k)$ is small. Second, $\mathbf{w}_k$ is a weak learner with $\mathrm{err}_{\mathcal{D}_{k-1}}(\mathbf{w}_k)$ at most a small constant. In our work, polynomial regression together with hinge loss minimization in the band guarantees both of these properties simultaneously by limiting its search to the halfspaces that are close in angle to $\mathbf{w}_{k-1}$ and limiting its distribution to $\mathcal{D}_{\mathbf{w}_{k-1}, \gamma_{k-1}}$. However, in the case of AVERAGE as we concentrate in the band $\mathcal{D}_{\mathbf{w}_{k-1}, \gamma_{k-1}}$ we bias the distributions towards its orthogonal component with respect to $\mathbf{w}_{k-1}$. Hence, an upper bound on $\theta(\mathbf{w}^*, \mathbf{w}_{k-1})$ only serves to assure that most of the data is orthogonal to $\mathbf{w}^*$ as well. Therefore, informally speaking, we lose the signal that otherwise could direct us in the direction of $\mathbf{w}^*$. More formally, consider the construction from Theorem 8.4.1 such that $\mathbf{w}_{k-1} = \mathbf{w}^* = (1, 0, \ldots, 0)$. In distribution $\mathcal{D}_{\mathbf{w}_{k-1}, \gamma_{k-1}}$, the component of $\mathbf{w}_k$ that is parallel to $\mathbf{w}_{k-1}$ scales down by the width of the band, γ_{k-1}. However, as most of the probability stays in a band passing through the origin in any log-concave (including Gaussian and uniform) distribution, the orthogonal component of $\mathbf{w}_k$ remains almost unchanged. Therefore, $\theta(\mathbf{w}_k, \mathbf{w}^*) = \theta(\mathbf{w}_k, \mathbf{w}_{k-1}) \in \Omega\left(\frac{1-\beta}{\gamma_{k-1}(1+\beta)}\right)$.

8.5 Hinge Loss Minimization Does Not Work

Hinge loss minimization is a widely used technique in machine learning. As we have shown in our work [24], which precedes the results discussed in Section 8.3, iteratively using hinge loss minimization alone within the band—instead of combining it with polynomial regression—can tolerate a very small amount of bounded noise (of the order of $\beta = 1 - 10^{-7}$). In this section, we show that a one-shot application of hinge loss minimization cannot learn halfspaces to arbitrary accuracy for any β. Indeed, our results go beyond hinge loss minimization and apply to one-shot optimization of other easy-to-optimize functions [25].

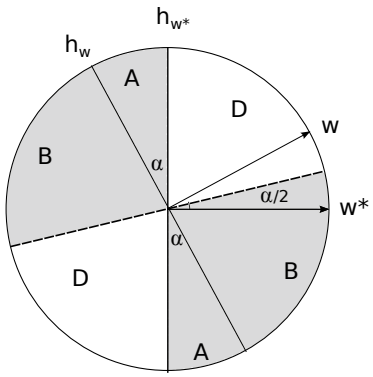


Figure 8.1: $\mathcal{D}_{\alpha,\beta}$

More concretely, we show that for every parameter τ and β, τ-hinge loss minimization does not recover the true halfspace on distributions with a uniform marginal over the unit ball in $\mathbb{R}^2$ and bounded noise with parameter β.

Theorem 8.5.1. *For every hinge-loss parameter $\tau \geq 0$ and every bounded noise parameter $0 \leq \beta < 1$ and true target halfspace $\mathbf{w}^*$, there is a distribution $\tilde{\mathcal{D}}$ with marginal that is uniform over the unit ball in $\mathbb{R}^2$ and bounded noise with parameter β, and there exists a constant c such that for*

$$\bar{\mathbf{w}} = \arg\min_{\mathbf{w}, \|\mathbf{w}\|_2 = 1} \mathbb{E}_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}}[\ell_\tau(\mathbf{w}, \mathbf{x}, y)],$$

we have $\theta(\bar{\mathbf{w}}, \mathbf{w}^*) > c$.

Note that since the theorem uses a distribution that has a marginal that is uniform over the unit ball in $\mathbb{R}^2$, $\theta(\bar{\mathbf{w}}, \mathbf{w}^*)$ is the disagreement between the two halfspaces. Using Lemma 8.2.1, the above theorem implies that the excess 0/1 error of the halfspace that minimizes the *expected* hinge loss is at least some constant. Since hinge loss is concentrated around its expectation, this implies that there exists a function $m(\epsilon) = O(\epsilon^{-2})$ such that for small enough excess error ϵ and for any $m > m(\epsilon)$ samples, with high probability the excess error of the empirical hinge loss minimizer is more than ϵ.

A remark is in order regarding the fact that we perform hinge loss minimization over *all unit vectors*. Our choice is due to the fact that, if $\mathbf{w}$ has lower hinge loss than $\mathbf{w}^*$, so does $\lambda\mathbf{w}$ compared to $\lambda\mathbf{w}^*$. More formally,

$$\ell_\tau(\lambda\mathbf{w}, \mathbf{x}, y) = \max\left(0,\ 1 - \frac{y(\lambda\mathbf{w}\cdot\mathbf{x})}{\tau}\right) = \max\left(0,\ 1 - \frac{y(\mathbf{w}\cdot\mathbf{x})}{\tau/\lambda}\right) = \ell_{\tau/\lambda}(\mathbf{w}, \mathbf{x}, y).$$

8.5.1 Proof of the Lower Bound

To prove the above result, we consider $\boldsymbol{\mathcal{P}}_\beta$ be the class of distributions with uniform marginal over the unit ball in $\mathbb{R}^2$ with β-bounded noise with respect to the Bayes classifier halfspace $\mathbf{w}^*$. We carefully design a subclass of $\boldsymbol{\mathcal{P}}_{\alpha,\beta} \subseteq \boldsymbol{\mathcal{P}}_\beta$, indexed by angle α and bounded noise parameter β as follows. Let Bayes optimal classifier be $h^* = h_{\mathbf{w}^*}$ for a unit vector $\mathbf{w}^*$. Let $h_{\mathbf{w}}$ be the

classifier that is defined by the unit vector $\mathbf{w}$ at angle α from $\mathbf{w}^*$. We partition the unit ball into areas A, B and D as in Figure 8.1. That is A consists of the two wedges of disagreement between $h_{\mathbf{w}}$ and $h_{\mathbf{w}^*}$ and the wedge where the two classifiers agree is divided in B (points that are closer to $h_{\mathbf{w}}$ than to $h_{\mathbf{w}^*}$) and D (points that are closer to $h_{\mathbf{w}^*}$ than to $h_{\mathbf{w}}$). We now flip the label of all points in areas A and B with probability $\eta = \frac{1-\beta}{2}$ and we do not introduce any noise in region D. More formally, points at angle between $\alpha/2$ and $\pi/2$ and points at angle between $\pi + \alpha/2$ and $-\pi/2$ from $\mathbf{w}^*$ are labeled with $h_{\mathbf{w}^*}(x)$ with probability 1. All other points are labeled $-h_{\mathbf{w}^*}(x)$ with probability η and $h_{\mathbf{w}^*}(x)$ with probability $(1-\eta)$. Clearly, this distribution satisfies the bounded noise condition with parameter β.

The goal of the above construction is to design distributions where $\mathbf{w}$ has smaller hinge loss than $\mathbf{w}^*$. We show that for every noise parameter β and hinge loss parameter τ, there is angle α, for which the expected hinge loss of $\mathbf{w}^*$ is more, by at least a constant, than that of $\mathbf{w}$ on $\tilde{\mathcal{D}}_{\alpha,\beta}$. Since hinge loss is continuous in angle of the halfspaces, this shows that no unit vector that is close in angle to $\mathbf{w}^*$ is the expected hinge loss minimizer either.

When considering the expected hinge loss of $\mathbf{w}$ and $\mathbf{w}^*$, note that

- noise in region A "evens out" the difference in hinge loss of $\mathbf{w}$ and $\mathbf{w}^*$. This is due to the fact that A is symmetric with respect to these two directions.
- noise in region B "helps $\mathbf{w}$". Since all points in region B are closer to the hyperplane defined by $\mathbf{w}$ than to the one defined by $\mathbf{w}^*$, vector $\mathbf{w}^*$ will pay more in hinge loss for the noise (and misclassification) in this region.
- there is no noise in region D. This is the region of points that are closer to the hyperplane defined by $\mathbf{w}^*$ than to the one defined by $\mathbf{w}$. Since both $\mathbf{w}^*$ and $\mathbf{w}$ classify the instances correctly, the cost for either classifier is small.

In the remainder of this section, let cA denote the hinge loss of $h_{\mathbf{w}^*}$ on one wedge (one half of) region A when all instance are labeled correctly and dA the hinge loss on the same region when all instance are labeled incorrectly. That is,

$$\mathrm{cA} = \frac{1}{\pi}\int_0^1 \int_0^\alpha (1 - \frac{z}{\tau}\sin(\varphi))z \, d\varphi \, dz,$$
$$\mathrm{dA} = \frac{1}{\pi}\int_0^1 \int_0^\alpha (1 + \frac{z}{\tau}\sin(\varphi))z \, d\varphi \, dz.$$

Moreover, let region C be the set of points at angle between $\pi - \alpha/2$ and $\pi + \alpha/2$ from $\mathbf{w}^*$ (See Figure 8.2), and let cC and dC be defined analogously to the above.

The following lemma described the relation between the expected hinge loss of $\mathbf{w}$ and $\mathbf{w}^*$. See Appendix B.1.

Lemma 8.5.2.

$$\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}_{\tau,\beta}}[\ell_\tau(\mathbf{w},\mathbf{x},y)] - \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}_{\tau,\beta}}[\ell_\tau(\mathbf{w}^*,\mathbf{x},y)] = \beta(\mathrm{dA} - \mathrm{cA}) - (1-\beta)(\mathrm{dC} - \mathrm{cC}).$$

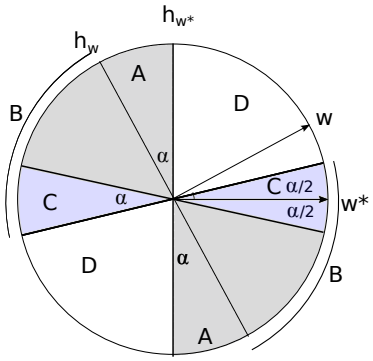


Figure 8.2: Area C

The next two lemmas evaluate $\mathrm{dA} - \mathrm{cA}$ and $\mathrm{dC} - \mathrm{cC}$. Their proofs are presented in Appendices B.2 and B.3.

Lemma 8.5.3. *Given $\tau_0 \geq 0$, let α be small enough such that region A is included in the τ_0-band around $\mathbf{w}^*$ (we can always choose the angle α sufficiently small for this). For all $\tau \geq \tau_0$:*

$$\mathrm{dA} - \mathrm{cA} = \frac{2}{3\pi\tau}(1 - \cos(\alpha)).$$

Lemma 8.5.4. *We have*

$$\mathrm{dC} - \mathrm{cC} \begin{cases} = \frac{4}{3\pi\tau}\sin\left(\frac{\alpha}{2}\right) & \text{when } \tau \geq 1, \\ \geq \frac{1}{\pi}\left(\frac{\alpha}{2} + \frac{2}{3\tau}\sin\left(\frac{\alpha}{2}\right) - \frac{\tau^2}{3}\tan\left(\frac{\alpha}{2}\right)\right) & \text{when } \tau < 1. \end{cases}$$

Proof of Theorem 8.5.1. We show that for any τ and β, there is α for which expected hinge loss of $\mathbf{w}$ is less than the expected hinge loss of $\mathbf{w}^*$ on distribution $\tilde{\mathcal{D}}_{\alpha,\beta}$. In particular, let α be such that

$$\text{Condition (1):} \quad \sin\left(\frac{\alpha}{2}\right) < \frac{1-\beta}{\beta} \quad \text{and}$$

$$\text{Condition (2):} \quad \frac{2(1-\cos(\alpha))}{\frac{\alpha\tau}{2} + \frac{2}{3}\sin\left(\frac{\alpha}{2}\right) - \frac{\tau^3}{3}\tan\left(\frac{\alpha}{2}\right)} < \frac{1-\beta}{\beta}.$$

Let us first show that such α indeed exists. For the first condition, note that $\sin\left(\frac{\alpha}{2}\right)$ is a monotone function with value 0 at $\alpha = 0$. So, for any β and τ, there is α_1 such that for all $\alpha \leq \alpha_1$, $\sin\left(\frac{\alpha}{2}\right) < (1-\beta)/\beta$. For the second condition note that

$$\lim_{\alpha\to 0} \frac{2(1-\cos(\alpha))}{\frac{\alpha\tau}{2} + \frac{2}{3}\sin\left(\frac{\alpha}{2}\right) - \frac{\tau^3}{3}\tan\left(\frac{\alpha}{2}\right)} = 0.$$

Moreover,

$$\lim_{\alpha\to 0} \frac{\partial}{\partial\alpha}\left(\frac{2(1-\cos(\alpha))}{\frac{\alpha\tau}{2} + \frac{2}{3}\sin\left(\frac{\alpha}{2}\right) - \frac{\tau^3}{3}\tan\left(\frac{\alpha}{2}\right)}\right) = \frac{6}{-2\tau^3 + 6\tau + 4} > 0.$$

Therefore, the above function is increasing and positive around 0. So, there is α_2, for which all $\alpha \leq \alpha_2$ satisfies condition (2). Therefore, $\alpha \leq \min\{\alpha_1, \alpha_2\}$ satisfies both conditions.

Next we show that for such a choice of α,

$$\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}_{\tau,\beta}}[\ell_\tau(\mathbf{w},\mathbf{x},y)] < \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\tilde{\mathcal{D}}_{\tau,\beta}}[\ell_\tau(\mathbf{w}^*,\mathbf{x},y)].$$

By Lemma 8.5.2, it suffices to show that

$$\frac{\mathrm{dA}-\mathrm{cA}}{\mathrm{dC}-\mathrm{cC}} < \frac{1-\beta}{\beta}.$$

For $\tau \geq 1$, using condition (1), we have

$$\begin{aligned}\frac{\mathrm{dA}-\mathrm{cA}}{\mathrm{dC}-\mathrm{cC}} &= \frac{1-\cos(\alpha)}{2\sin\left(\frac{\alpha}{2}\right)} \\ &= \sin\left(\frac{\alpha}{2}\right) \\ &< \frac{1-\beta}{\beta},\end{aligned}$$

as desired. For $\tau < 1$, using condition (2), we have

$$\begin{aligned}\frac{\mathrm{dA}-\mathrm{cA}}{\mathrm{dC}-\mathrm{cC}} &\leq \frac{2(1-\cos(\alpha))}{\frac{\alpha\tau}{2}+\frac{2}{3}\sin\left(\frac{\alpha}{2}\right)-\frac{\tau^3}{3}\tan\left(\frac{\alpha}{2}\right)} \\ &< \frac{1-\beta}{\beta}.\end{aligned}$$

This proves the theorem. □

8.6 Discussion and Subsequent Works

Our work presented in this chapter is the first to provide a computationally efficient algorithm that is robust to the presence of bounded noise. Several aspects of bounded noise make it such an interesting noise model to study. First, it is a natural model that captures noise that may appear in crowd-sourced data sets. Second, it is a natural middleground between the fully adversarial noise model, where obtaining arbitrarily small excess error is hard even in restricted settings, and the very stylized random classification noise, where multiple efficient algorithms are known to be effective. Our results extend the learning guarantees of obtaining highly accurate classifiers—that was previously only known in the random classification noise—to bounded noise (under log-concave distributions). Third, bounded noise is a well-studied distributional assumption that has been identified in statistical learning to yield fast statistical rates of convergence. While both computational and statistical efficiency are crucial in machine learning applications, computational and statistical complexity have been studied under disparate sets of assumptions and models. Our results can be viewed as a step towards bringing these two lines of research closer together.

8.6.1 Subsequent Works

Following the initial publication of the results that appear in this chapter, subsequent efforts have been made to extend and strengthen these results. Alongside some of the results presented in this chapter, we [25] also consider the problem of learning halfspaces in presence of bounded noise in the context of *attribute-efficient learning*. In this setting, the assumption is that the true halfspace $\mathbf{w}^*$ is sparse, i.e., $\|\mathbf{w}^*\|_0 \leq t$, and the goal is to learn a classifier within $\mathrm{poly}(t, \ln(d))$ number of samples. In our work [25], excluded from this dissertation, we show that a variant of Algorithm 8.1 learns a halfspace with excess error ϵ using a number of labeled or unlabeled samples that is $\mathrm{poly}\left(t, \frac{1}{\epsilon}, \ln(d)\right)$.

Yan and Zhang [269] studies a variant of the margin-based approach where a variant of Perceptron is used within the band, rather than a combination of polynomial regression and hinge loss minimization. Their algorithm requires $O\left(\frac{d}{\beta^2}\right)$ samples when the marginal distribution is uniform over the unit ball, which has a better dependence on noise parameter β than our $O\left(d^{1/\beta^4}\right)$ sample complexity. Subsequently, Zhang [272] showed that when $\mathbf{w}^*$ is sparse, the number of required labeled samples can be additionally improved to $\mathrm{poly}\left(t, \ln\left(\frac{1}{\epsilon}\right), \ln(d)\right)$ for general log-concave distributions.

Chapter 9

Efficient PAC Learning from the Crowd

9.1 Introduction

Over the last decade, research in machine learning and AI has seen tremendous growth, partly due to the ease with which we can collect and annotate massive amounts of data across various domains. This rate of data annotation has been made possible due to crowdsourcing tools, such as Amazon Mechanical Turk™, that facilitate individuals' participation in a labeling task. In the context of classification, a crowdsourced model uses a large pool of workers to gather labels for a given training data set that will be used for the purpose of learning a good classifier. Such learning environments that involve the crowd give rise to a multitude of design choices that do not appear in traditional learning environments. These include: How does the goal of learning from the crowd differs from the goal of annotating data by the crowd? What challenges does the high amount of noise typically found in curated data sets [164, 179, 267] pose to the learning algorithms? How do learning and labeling processes interplay? How many labels are we willing to take per example? And, how much load can a labeler handle?

In recent years, there have been many exciting works addressing various theoretical aspects of these and other questions [250], such as reducing noise in crowdsourced data [98], task assignment [30, 258] in online or offline settings [174], and the role of incentives [160]. In this chapter we focus on one such aspect, namely, *how to efficiently learn and generalize from the crowd with minimal cost?* The standard approach is to view the process of acquiring labeled data through crowdsourcing and the process of learning a classifier in isolation. In other words, a typical learning process involves collecting data labeled by many labelers via a crowdsourcing platform followed by running a passive learning algorithm to extract a good hypothesis from the labeled data. As a result, approaches to crowdsourcing focus on getting high quality labels per example and not so much on the task further down in the pipeline. Naive techniques such as taking majority votes to obtain almost perfect labels have a cost per labeled example that scales with the data size, namely $\log(\frac{m}{\delta})$ queries per label where m is the training data size and δ is the desired failure probability. This is undesirable in many scenarios when data size is large. Furthermore, if only a small fraction of the labelers in the crowd are perfect, such approaches will inevitably fail. An alternative is to feed the noisy labeled data to existing passive learning algorithms. However, we currently lack computationally efficient PAC learning algorithms that

are provably robust to high amounts of noise that exists in crowdsourced data. Hence separating the learning process from the data annotation process results in high labeling costs or suboptimal learning algorithms.

In light of the above, we initiate the study of designing efficient PAC learning algorithms in a crowdsourced setting where learning and acquiring labels are done in tandem. We consider a natural model of crowdsourcing and ask the fundamental question of whether efficient learning with little overhead in labeling cost is possible in this scenario. We focus on the classical PAC setting of Valiant [262] where there exists a true target classifier $f^* \in \mathcal{F}$ and the goal is to learn $\mathcal{F}$ from a finite training set generated from the underlying distribution. We assume that one has access to a large pool of labelers that can provide (noisy) labels for the training set. We seek algorithms that run in polynomial time and produce a hypothesis with small error. We are especially interested in settings where there are computationally efficient algorithms for learning $\mathcal{F}$ in the consistency model, i.e. the realizable PAC setting. Additionally, we also want our algorithms to make as few label queries as possible, ideally requesting a total number of labels that is within a constant factor of the amount of labeled data needed in the realizable PAC setting. We call this $O(1)$ *overhead* or *cost per labeled example*. Furthermore, in a realistic scenario each labeler can only provide labels for a constant number of examples, hence we cannot ask too many queries to a single labeler. We call the number of queries asked to a particular labeler the *load* of that labeler.

Perhaps surprisingly, we show that when a noticeable fraction of the labelers in our pool are *perfect* all of the above objectives can be achieved simultaneously. That is, *if $\mathcal{F}$ can be efficiently PAC learned in the realizable PAC model, then it can be efficiently PAC learned in the noisy crowdsourcing model with a constant cost per labeled example*. In other words, the ratio of the number of label requests in the noisy crowdsourcing model to the number of labeled examples needed in the traditional PAC model with a perfect labeler is a constant and does not increase with the size of the data set. Additionally, each labeler is asked to label only a constant number of examples, i.e., $O(1)$ load per labeler. Our results also answer an open question of Dekel and Shamir [98] regarding the possibility of efficient noise robust PAC learning by performing labeling and learning simultaneously. When no perfect labelers exist, a related task is to find a set of the labelers which are *good* but not perfect. We show that we can identify the set of all good labelers, when at least the majority of labelers are good.

9.1.1 Overview of Results

We study various versions of the model described above. In the most basic setting we assume that a large percentage, say 70% of the labelers are *perfect*, i.e., they always label according to the target function f^*. The remaining 30% of the labelers could behave arbitrarily and we make no assumptions on them. Since the perfect labelers are in strong majority, a straightforward approach is to label each example with the majority vote over a few randomly chosen labelers, to produce the correct label on every instance with high probability. However, such an approach leads to a query bound of $O(\log \frac{m}{\delta})$ per labeled example, where m is the size of the training set and δ is the acceptable probability of failure. In other words, the cost per labeled example is $O(\log \frac{m}{\delta})$ and scales with the size of the data set. Another easy approach is to pick a few labelers at random and ask them to label all the examples. Here, the cost per labeled example is a

constant but the approach is infeasible in a crowdsourcing environment since it requires a single or a constant number of labelers to label the entire data set. Yet another approach is to label each example with the majority vote of $O(\log \frac{1}{\epsilon})$ labelers. While the labeled sample set created in this way only has error of ϵ, it is still unsuitable for being used with PAC learning algorithms as they are not robust to even small amounts of noise, if the noise is heterogeneous. So, the computational challenges still persist. Nevertheless, we introduce an algorithm that performs *efficient* learning with $O(1)$ *cost per labeled example* and $O(1)$ *load per labeler.*

Theorem 9.4.3 (Informal) *Let $\mathcal{F}$ be a hypothesis class that can be PAC learned in polynomial time to ϵ error with probability $1-\delta$ using $m_{\epsilon,\delta}$ samples. Then $\mathcal{F}$ can be learned in polynomial time using $O(m_{\epsilon,\delta})$ samples in a crowdsourced setting with $O(1)$ cost per labeled example, provided a $\frac{1}{2}+\Theta(1)$ fraction of the labelers are perfect. Furthermore, every labeler is asked to label only 1 example.*

Notice that the above theorem immediately implies that each example is queried only $O(1)$ times on average as opposed to the data size dependent $O(\log(\frac{m}{\delta}))$ cost incurred by the naive majority vote style procedures. We next extend our result to the setting where the fraction of perfect labelers is significant but might be less than $\frac{1}{2}$, say 0.4. Here we again show that $\mathcal{F}$ can be efficiently PAC learned using $O(m_{\epsilon,\delta})$ queries provided we have access to an "expert" that can correctly label a constant number of examples. We call such queries that are made to an expert *golden queries*. When the fraction of perfect labelers is close to $\frac{1}{2}$, say 0.4, we show that *just one* golden query is enough to learn. More generally, when the fraction of the perfect labelers is some α, we show that $O(1/\alpha)$ golden queries is sufficient to learn a classifier efficiently. We describe our results in terms of α, but we are particularly interested in regimes where $\alpha = \Theta(1)$.

Theorem 9.4.13 (Informal) *Let $\mathcal{F}$ be a hypothesis class that can be PAC learned in polynomial time to ϵ error with probability $1-\delta$ using $m_{\epsilon,\delta}$ samples. Then $\mathcal{F}$ can be learned in polynomial time using $O(m_{\epsilon,\delta})$ samples in a crowdsourced setting with $O(\frac{1}{\alpha})$ cost per labeled example, provided more than an α fraction of the labelers are perfect for some constant $\alpha > 0$. Furthermore, every labeler is asked to label only $O(\frac{1}{\alpha})$ examples and the algorithm uses at most $\frac{2}{\alpha}$ golden queries.*

The above two theorems highlight the importance of incorporating the structure of the crowd in algorithm design. Being oblivious to the labelers will result in noise models that are notoriously hard. For instance, if one were to assume that each example is labeled by a single random labeler drawn from the crowd, one would recover the *Malicious Misclassification Noise* of Rivest and Sloan [231]. Getting computationally efficient learning algorithms even for very simple hypothesis classes has been a long standing open problem in this space. Our results highlight that by incorporating the structure of the crowd, one can efficiently learn *any hypothesis class* with a small overhead.

Finally, we study the scenario when none of the labelers are perfect. Here we assume that the majority of the labelers are "good", that is they provide labels according to functions that are all ϵ-close to the target function. In this scenario generating a hypothesis of low error is as hard as agnostic learning[1]. Nonetheless, we show that one can detect all of the good labelers using

[1]This can happen for instance when all the labelers label according to a single function f that is ϵ-far from f^*.

expected $O(\frac{1}{\epsilon}\log(n))$ queries per labeler, where n is the target number of labelers desired in the pool.

Theorem 9.5.1 (Informal) *Assume we have a target set of n labelers that are partitioned into two sets,* good *and* bad*. Furthermore, assume that there are at least $\frac{n}{2}$ good labelers who always provide labels according to functions that are ϵ-close to a target function f^*. The set of bad labelers always provide labels according to functions that are at least 4ϵ away from the target. Then there is a polynomial time algorithm that identifies, with probability at least $1-\delta$, all the good labelers and none of the bad labelers using expected $O(\frac{1}{\epsilon}\log(\frac{n}{\delta}))$ queries per labeler.*

9.1.2 Related Work

Crowdsourcing has received significant attention in the machine learning community. As mentioned in the introduction, crowdsourcing platforms require one to address several questions that are not present in traditional modes of learning.

The work of Dekel and Shamir [98] shows how to use crowdsourcing to reduce the noise in a training set before feeding it to a learning algorithm. Our results answer an open question in their work by showing that performing data labeling and learning in tandem can lead to significant benefits.

A large body of work in crowdsourcing has focused on the problem of *task assignment*. Here, workers arrive in an online fashion and a requester has to choose to assign specific tasks to specific workers. Additionally, workers might have different abilities and might charge differently for the same task. The goal from the requester's point of view is to finish multiple tasks within a given budget while maintaining a certain minimum quality [160, 258]. There is also significant work on *dynamic procurement* where the focus is on assigning prices to the given tasks so as to provide incentive to the crowd to perform as many of them as possible within a given budget [29, 30, 248]. Unlike our setting, the goal in these works is not to obtain a generalization guarantee or learn a function, but rather to complete as many tasks as possible within the budget.

The work of Karger et al. [173, 174] also studies the problem of task assignment in offline and online settings. In the offline setting, the authors provide an algorithm based on belief propagation that infers the correct answers for each task by pooling together the answers from each worker. They show that their approach performs better than simply taking majority votes. Unlike our setting, their goal is to get an approximately correct set of answers for the given data set and not to generalize from the answers. Furthermore, their model assumes that each labeler makes an error at random independently with a certain probability. We, on the other hand, make no assumptions on the nature of the *bad* labelers.

Another related model is the recent work of Steinhardt et al. [253]. Here the authors look at the problem of extracting top rated items by a group of labelers among whom a constant fraction are consistent with the *true* ratings of the items. The authors use ideas from matrix completion to design an algorithm that can recover the top rated items with an ϵ fraction of the noise provided every labeler rates $\approx \frac{1}{\epsilon^4}$ items and one has access to $\approx \frac{1}{\epsilon^2}$ ratings from a trusted expert. Their model is incomparable to ours since their goal is to recover the top rated items and not to learn a hypothesis that generalizes to a test set.

Our results also shed insights into the notorious problem of PAC learning with noise. Despite decades of research into PAC learning, noise tolerant polynomial time learning algorithms remain elusive. There has been substantial work on PAC learning under realistic noise models such as the Massart noise or the Tsybakov noise models [65]. However, computationally efficient algorithms for such models are known in very restricted cases [24, 25]. In contrast, we show that by using the structure of the crowd, one can indeed design polynomial time PAC learning algorithms even when the noise is of the type mentioned above.

More generally, interactive models of learning have been studied in the machine learning community. The most popular among them is the area of active learning [34, 82, 92, 148, 182]. In this model, the learning algorithm can adaptively query for the labels of a few examples in the training set and use them to produce an accurate hypothesis. The goal is to use as few label queries as possible. The number of labeled queries used is called the *label complexity* of the algorithm. It is known that certain hypothesis classes can be learned in this model using much fewer labeled queries than predicted by the VC theory. In particular, in many instances the label complexity scales only logarithmically in $\frac{1}{\epsilon}$ as opposed to linearly in $\frac{1}{\epsilon}$. However, to achieve computational efficiency, the algorithms in this model rely on the fact that one can get perfect labels for every example queried. This would be hard to achieve in our model since in the worst case it would lead to each labeler answering $\log(\frac{d}{\epsilon})$ many queries. In contrast, we want to keep the query load of a labeler to a constant and hence the techniques developed for active learning are insufficient for our purposes. Furthermore, in noisy settings most work on efficient active learning algorithms assumes the existence of an *empirical risk minimizer* (ERM) oracle that can minimize training error even when the instances aren't labeled according to the target classifier. However, in most cases such an ERM oracle is hard to implement and the improvements obtained in the label complexity are less drastic in such noisy scenarios.

Another line of work initiated by Zhang and Chaudhuri [273] models related notions of weak and strong labelers in the context of active learning. The authors study scenarios where the label queries to the strong labeler can be reduced by querying the weak and potentially noisy labelers more often. However, as discussed above, the model does not yield relevant algorithms for our setting as in the worst case one might end up querying for $\frac{d}{\epsilon}$ high quality labels leading to a prohibitively large load per labeler in our setting. The work of Yan et al. [270] studies a model of active learning where the labeler abstains from providing a label prediction more often on instances that are closer to the decision boundary. The authors then show how to use the abstentions in order to approximate the decision boundary. Our setting is inherently different, since we make no assumptions on the bad labelers.

9.2 Model and Notations

Let $\mathcal{X}$ be an instance space and $\mathcal{Y} = \{+1, -1\}$ be the set of possible labels. A *hypothesis* is a function $f : \mathcal{X} \to \mathcal{Y}$ that maps an instance $x \in \mathcal{X}$ to its classification y. We consider the *realizable* setting where there is a distribution over $\mathcal{X} \times \mathcal{Y}$ and a true target function in hypothesis class $\mathcal{F}$. More formally, we consider a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and an unknown hypothesis $f^* \in \mathcal{F}$, where $\mathrm{err}_{\mathcal{D}}(f^*) = 0$. We denote the marginal of $\mathcal{D}$ over $\mathcal{X}$ by $\mathcal{D}_{|\mathcal{X}}$. The *error* of a hypothesis f with respect to distribution $\mathcal{D}$ is defined as $\mathrm{err}_{\mathcal{D}}(f) = \Pr_{(x,f^*(x))\sim\mathcal{D}}[f(x) \neq f^*(x)]$.

In order to achieve our goal of learning f^* well with respect to distribution $\mathcal{D}$, we consider having access to a large pool of labelers, some of whom label according to f^* and some who do not. Formally, labeler i is defined by its corresponding classification function $g_i : \mathcal{X} \to \mathcal{Y}$. We say that g_i is *perfect* if $\mathrm{err}_{\mathcal{D}}(g_i) = 0$. We consider a distribution P that is uniform over all labelers and let $\alpha = \Pr_{i\sim P}[\mathrm{err}_{\mathcal{D}}(g_i) = 0]$ be the fraction of perfect labelers. We allow an algorithm to query labelers on instances drawn from $\mathcal{D}_{|\mathcal{X}}$. Our goal is to design learning algorithms that efficiently learn a low error classifier while maintaining a small overhead in the number of labels. We compare the computational and statistical aspects of our algorithms to their PAC counterparts in the realizable setting.

In the traditional PAC setting with a realizable distribution, $m_{\epsilon,\delta}$ denotes the number of samples needed for learning $\mathcal{F}$. That is, $m_{\epsilon,\delta}$ is the total number of labeled samples drawn from the realizable distribution $\mathcal{D}$ needed to output a classifier f that has $\mathrm{err}_{\mathcal{D}}(f) \leq \epsilon$, with probability $1-\delta$. We know from the VC theory [11], that for a hypothesis class $\mathcal{F}$ with VC-dimension d and no additional assumptions on $\mathcal{F}$, $m_{\epsilon,\delta} \in O\left(\epsilon^{-1}\left(d\ln\left(\frac{1}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right)\right)$. Furthermore, we assume that efficient algorithms for the realizable setting exist. That is, we consider an oracle $\mathcal{O}_{\mathcal{F}}$ that for a set of labeled instances S, returns a function $f \in \mathcal{F}$ that is consistent with the labels in S, if one such function exists, and outputs "None" otherwise.

Given an algorithm in the noisy crowd-sourcing setting, we define the *average cost per labeled example* of the algorithm, denoted by Λ, to be the ratio of the number of label queries made by the algorithm to the number of labeled examples needed in the traditional realizable PAC model, $m_{\epsilon,\delta}$. The *load* of an algorithm, denoted by λ, is the *maximum number of label queries that have to be answered by an individual labeler.* In other words, λ is the maximum number of labels queried from one labeler, when P has an infinitely large support. [2] When the number of labelers is fixed, such as in Section 9.5, we define the *load* to simply be the number of queries answered by a single labeler. Moreover, we allow an algorithm to directly query the target hypothesis f^* on a few, e.g., $O(1)$, instances drawn from $\mathcal{D}_{|\mathcal{X}}$. We call these "golden queries" and denote their total number by Γ.

Given a set of labelers L and an instance $x \in \mathcal{X}$, we define $\mathrm{Maj}_L(x)$ to be the label assigned to x by the majority of labelers in L. Moreover, we denote by $\text{Maj-size}_L(x)$ the fraction of the labelers in L that agree with the label $\mathrm{Maj}_L(x)$. Given a set of classifiers H, we denote by $MAJ(H)$ the classifier that for each x returns prediction $\mathrm{Maj}_H(x)$. Given a distribution P over labelers and a set of labeled examples S, we denote by $P_{|S}$ the distribution P conditioned on labelers that agree with labeled samples $(x, y) \in S$. We consider S to be small, typically of size $O(1)$. Note that we can draw a labeler from $P_{|S}$ by first drawing a labeler according to P and querying it on all the labeled instances in S. Therefore, when P has infinitely large support, the load of an algorithm is the maximum size of S that P is ever conditioned on.

[2] The concepts of *total number of queries* and *load* may be seen as analogous to *work* and *depth* in parallel algorithms, where *work* is the total number of operations performed by an algorithm and *depth* is the maximum number of operations that one processor has to perform in a system with infinitely many processors.

9.3 A Baseline Algorithm and a Road-map for Improvement

In this section, we briefly describe a simple algorithm and the approach we use to improve over it. Consider a very simple baseline algorithm for the case of $\alpha > \frac{1}{2}$:

> **BASELINE**: Draw a sample of size $m = m_{\epsilon,\delta}$ from $\mathcal{D}_{|\mathcal{X}}$ and label each $x \in S$ by $\mathrm{Maj}_L(x)$, where $L \sim P^k$ for $k = O\left((\alpha - 0.5)^{-2} \ln\left(\frac{m}{\delta}\right)\right)$ is a set of randomly drawn labelers. Return classifier $\mathcal{O}_{\mathcal{F}}(S)$.

That is, the baseline algorithm queries enough labelers on each sample such that with probability $1 - \delta$ all the labels are correct. Then, it learns a classifier using this labeled set. It is clear that the performance of BASELINE is far from being desirable. First, this approach takes $\log(m/\delta)$ more labels than it requires samples, leading to an average cost per labeled example that increases with the size of the sample set. Moreover, when perfect labelers form a small majority of the labelers, i.e., $\alpha = \frac{1}{2} + o(1)$, the number of labels needed to correctly label an instance increases drastically. Perhaps even more troubling is that if the perfect labelers are in minority, i.e., $\alpha < \frac{1}{2}$, S may be mislabeled and $\mathcal{O}_{\mathcal{F}}(S)$ may return a classifier that has large error, or no classifier at all. In this work, we improve over BASELINE in both aspects.

In Section 9.4, we improve the $\log(m/\delta)$ average cost per labeled example by interleaving the two processes responsible for learning a classifier and querying labels. In particular, BASELINE first finds *high quality labels*, i.e., labels that are correct with high probability, and then learns a classifier that is consistent with those labeled samples. However, interleaving the process of learning and acquiring high quality labels can make both processes more efficient. At a high level, for a given classifier h that has a larger than desirable error, one may be able to find regions where h performs particularly poorly. That is, the classifications provided by h may differ from the correct label of the instances. In turn, by focusing our effort for getting high quality labels on these regions we can output a correctly labeled sample set using less label queries overall. These additional correctly labeled instances from regions where h performs poorly can help us improve the error rate of h in return. In Section 9.4, we introduce an algorithm that draws on ideas from boosting and a probabilistic filtering approach that we develop in this work to facilitate interactions between learning and querying.

In Section 9.4.1, we remove the dependence of label complexity on $(\alpha - 0.5)^{-2}$ using $O(1/\alpha)$ golden queries. At a high level, instances where only a small majority of labelers agree are difficult to label using queries asked from labelers. But, these instances are great test cases that help us identify a large fraction of imperfect labelers. That is, we can first ask a golden query on one such instance to get its correct label and from then on only consider labelers that got this label correctly. In other words, we first test the labelers on one or very few tests questions, if they pass the tests, then we ask them real label queries for the remainder of the algorithm, if not, we never consider them again.

9.4 An Interleaving Algorithm

In this section, we improve over the average cost per labeled example of the BASELINE algorithm, by interleaving the process of learning and acquiring high quality labels. Our Algorithm 9.2

facilitates the interactions between the learning process and the querying process using ideas from classical PAC learning and adaptive techniques we develop in this work. For ease of presentation, we first consider the case where $\alpha = \frac{1}{2} + \Theta(1)$, say $\alpha \geq 0.7$, and introduce an algorithm and techniques that work in this regime. In Section 9.4.1, we show how our algorithm can be modified to work with any value of α. For convenience, we assume in the analysis below that distribution $\mathcal{D}$ is over a discrete space. This is in fact without loss of generality, since using uniform convergence one can instead work with the uniform distribution over an unlabeled sample multiset of size $O(\frac{d}{\epsilon^2})$ drawn from $\mathcal{D}_{|\mathcal{X}}$.

Here, we provide an overview of the techniques and ideas used in this algorithm.

Boosting: In general, boosting algorithms [122, 123, 243] provide a mechanism for producing a classifier of error ϵ using learning algorithms that are only capable of producing classifiers with considerably larger error rates, typically of error $p = \frac{1}{2} - \gamma$ for small γ. In particular, early work of Schapire [243] in this space shows how one can combine 3 classifiers of error p to get a classifier of error $3p^2 - 2p^3$, for any $p > 0$.

Theorem 9.4.1 (Schapire [243])**.** *For any $p > 0$ and distribution $\mathcal{D}$, consider three classifiers: 1) classifier h_1 such that $\mathrm{err}_{\mathcal{D}}(h_1) \leq p$; 2) classifier h_2 such that $\mathrm{err}_{\mathcal{D}_2}(h_2) \leq p$, where $\mathcal{D}_2 = \frac{1}{2}\mathcal{D}_C + \frac{1}{2}\mathcal{D}_I$ for distributions $\mathcal{D}_C$ and $\mathcal{D}_I$ that denote distribution $\mathcal{D}$ conditioned on $\{x \mid h_1(x) = f^*(x)\}$ and $\{x \mid h_1(x) \neq f^*(x)\}$, respectively; 3) classifier h_3 such that $\mathrm{err}_{\mathcal{D}_3}(h_3) \leq p$, where $\mathcal{D}_3$ is $\mathcal{D}$ conditioned on $\{x \mid h_1(x) \neq h_2(x)\}$. Then, $\mathrm{err}_{\mathcal{D}}(MAJ(h_1, h_2, h_3)) \leq 3p^2 - 2p^3$.*

As opposed to the main motivation for boosting where the learner only has access to a learning algorithm of error $p = \frac{1}{2} - \gamma$ for a small value of γ, in our setting we can learn a classifier to *any desired error rate* p as long as we have a sample set of $m_{p,\delta}$ correctly labeled instances. The larger the error rate p, the smaller the total number of label queries needed for producing a correctly labeled set of the appropriate size. We use this idea in Algorithm 9.2. In particular, we learn classifiers of error $p = O(\sqrt{\epsilon})$ using sample sets of size $O(m_{\sqrt{\epsilon},\delta})$ that are labeled by majority vote of $O(\log(m_{\sqrt{\epsilon},\delta}))$ labelers, using fewer label queries overall than BASELINE. We then use Theorem 9.4.1 to produce a classifier of error $3p^2 - 2p^3$ that is $O(\epsilon)$ for small enough values of ϵ.

Probabilistic Filtering: Given classifier h_1, the second step of the classical boosting algorithm requires distribution $\mathcal{D}$ to be reweighed based on the correctness of h_1. This step can be done by a *filtering* process as follows: Take a large set of labeled samples from $\mathcal{D}$ and divide them to two sets depending on whether or not the instances are mislabeled by h_1. Distribution $\mathcal{D}_2$, in which instances mislabeled by h_1 make up half of the weight, can be simulated by picking each set with probability $\frac{1}{2}$ and taking an instance from that set uniformly at random. To implement *filtering* in our setting, however, we would need to first get high quality labels for the set of instances used for simulating $\mathcal{D}_2$. Furthermore, this sample set is typically large, since at least $\frac{1}{p}m_{p,\delta}$ random samples from $\mathcal{D}$ are needed to simulate $\mathcal{D}_2$ that has half of its weight on the points that h_1 mislabels (which is a p fraction of the total points). In our case where $p = O(\sqrt{\epsilon})$, getting high quality labels for such a large sample set requires $O\left(m_{\epsilon,\delta} \ln\left(\frac{m_{\epsilon,\delta}}{\delta}\right)\right)$ label queries, which is as large as the total number of labels queried by BASELINE.

Algorithm 9.1: FILTER(S, h)

Let $S_I = \emptyset$ and $N = \log\left(\frac{1}{\epsilon}\right)$.
for $x \in S$ **do**
  for $t = 1, \ldots, N$ **do**
    Draw a random labeler $i \sim P$ and let $y_t = g_i(x)$
    if t is odd and $\mathrm{Maj}(y_{1:t}) = h(x)$ **then** break **end if**
  end for
  Let $S_I = S_I \cup \{x\}$. // Reaches this step when for all t, $\mathrm{Maj}(y_{1:t}) \neq h(x)$
end for
return S_I

In this work, we introduce a *probabilistic filtering* approach, called FILTER, that only requires $O\left(m_{\epsilon,\delta}\right)$ label queries, i.e., $O(1)$ cost per labeled example. Given classifier h_1 and an unlabeled sample set S, FILTER(S, h_1) returns a set $S_I \subseteq S$ such that *for any $x \in S$ that is mislabeled by h_1, $x \in S_I$ with probability at least $\Theta(1)$. Moreover, any x that is correctly labeled by h_1 is most likely not included in S_I.* This procedure is described in detail in Algorithm 9.1. Here, we provide a brief description of its working: For any $x \in S$, FILTER queries one labeler at a time, drawn at random, until the majority of the labels it has acquired so far agree with $h_1(x)$, at which point FILTER removes x from consideration. On the other hand, if the majority of the labels never agree with $h_1(x)$, FILTER adds x to the output set S_I. Consider $x \in S$ that is correctly labeled by h. Since each additional label agrees with $h_1(x) = f^*(x)$ with probability ≥ 0.7, with high probability the majority of the labels on x will agree with $f^*(x)$ at some point, in which case FILTER stops asking for more queries and removes x. As we show in Lemma 9.4.9 this happens within $O(1)$ queries most of the time. On the other hand, for x that is mislabeled by h, a labeler agrees with $h_1(x)$ with probability ≤ 0.3. Clearly, for one set of random labelers —one snapshot of the labels queried by FILTER— the majority label agrees with $h_1(x)$ with a very small probability. As we show in Lemma 9.4.6, even when considering the progression of all labels queried by FILTER throughout the process, with probability $\Theta(1)$ the majority label never agrees with $h_1(x)$. Therefore, x is added to S_I with probability $\Theta(1)$.

Super-sampling: Another key technique we use in this work is *super-sampling*. In short, this means that as long as we have the correct label of the sampled points and we are in the realizable setting, more samples never hurt the algorithm. Although this seems trivial at first, it does play an important role in our approach. In particular, our probabilistic filtering procedure does not necessarily simulate $\mathcal{D}_2$ but a distribution $\mathcal{D}'$, such that $\Theta(1)d_2(x) \leq d'(x)$ for all x, where d_2 and d' are the densities of $\mathcal{D}_2$ and $\mathcal{D}'$, respectively. At a high level, sampling $\Theta(m)$ instances from $\mathcal{D}'$ simulates a super-sampling process that samples m instances from $\mathcal{D}_2$ and then adds in some arbitrary instances.

Lemma 9.4.2. *Given a hypothesis class $\mathcal{F}$ consider any two discrete distributions $\mathcal{D}$ and $\mathcal{D}'$ such that for all x, $d'(x) \geq c \cdot d(x)$ for an absolute constant $c > 0$, and both distributions are labeled according to $f^* \in \mathcal{F}$. There exists a constant $c' > 1$ such that for any ϵ and δ, with*

probability $1-\delta$ *over a labeled sample set* S *of size* $c'm_{\epsilon,\delta}$ *drawn from* $\mathcal{D}'$*,* $\mathcal{O}_{\mathcal{F}}(S)$ *has error of at most* ϵ *with respect to distribution* $\mathcal{D}$*.*

Proof. First, notice that because $\mathcal{D}$ and $\mathcal{D}'$ are both labeled according to $f^* \in \mathcal{F}$, for any $f \in \mathcal{F}$ we have,

$$\operatorname*{err}_{\mathcal{D}'}(f) = \sum_x d'(x)\mathbb{1}_{f(x)\neq f^*(x)} \geq \sum_x c \cdot d(x)\mathbb{1}_{f(x)\neq f^*(x)} = c \cdot \operatorname*{err}_{\mathcal{D}}(f).$$

Therefore, if $\operatorname{err}_{\mathcal{D}'}(f) \leq c\epsilon$, then $\operatorname{err}_{\mathcal{D}}(f) \leq \epsilon$. Let $m' = m_{c\epsilon,\delta}$, we have

$$\begin{aligned}\delta &> \Pr_{S'\sim\mathcal{D}'^{m'}}[\exists f \in \mathcal{F}, \text{s.t. } \operatorname*{err}_{S'}(f) = 0 \wedge \operatorname*{err}_{\mathcal{D}'}(f) \geq c\epsilon] \\ &\geq \Pr_{S'\sim\mathcal{D}'^{m'}}[\exists f \in \mathcal{F}, \text{s.t. } \operatorname*{err}_{S'}(f) = 0 \wedge \operatorname*{err}_{\mathcal{D}}(f) \geq \epsilon].\end{aligned}$$

The claim follows by the fact that $m_{c\epsilon,\delta} = O\left(\frac{1}{c}m_{\epsilon,\delta}\right)$. □

With these techniques at hand, we present Algorithm 9.2. At a high level, the algorithm proceeds in three phases, one for each classifier used by Theorem 9.4.1. In Phase 1, the algorithm learns h_1 such that $\operatorname{err}_{\mathcal{D}}(h_1) \leq \frac{1}{2}\sqrt{\epsilon}$. In Phase 2, the algorithm first filters a set of size $O(m_{\epsilon,\delta})$ into the set S_I and takes an additional set S_C of $\Theta(m_{\sqrt{\epsilon},\delta})$ samples. Then, it queries $O(\log(\frac{m_{\epsilon,\delta}}{\delta}))$ labelers on each instance in S_I and S_C to get their correct labels with high probability. Next, it partitions these instances to two different sets based on whether or not h_1 made a mistake on them. It then learns h_2 on a sample set $\overline{W}$ that is drawn by weighting these two sets equally. As we show in Lemma 9.4.8, $\operatorname{err}_{\mathcal{D}_2}(h_2) \leq \frac{1}{2}\sqrt{\epsilon}$. In phase 3, the algorithm learns h_3 on a sample set S_3 drawn from $\mathcal{D}_{|\mathcal{X}}$ conditioned on h_1 and h_2 disagreeing. Finally, the algorithm returns $MAJ(h_1, h_2, h_3)$.

Theorem 9.4.3 ($\boldsymbol{\alpha = \frac{1}{2} + \Theta(1)}$ case)**.** *Algorithm 9.2 uses oracle* $\mathcal{O}_{\mathcal{F}}$*, runs in time* $\operatorname{poly}(d, \frac{1}{\epsilon}, \ln(\frac{1}{\delta}))$ *and with probability* $1-\delta$ *returns* $f \in \mathcal{F}$ *with* $\operatorname{err}_{\mathcal{D}}(f) \leq \epsilon$*, using* $\Lambda = O\left(\sqrt{\epsilon}\log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right) + 1\right)$ *cost per labeled example,* $\Gamma = 0$ *golden queries, and* $\lambda = 1$ *load. Note that when* $\frac{1}{\sqrt{\epsilon}} \geq \log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right)$*, the above cost per labeled sample is* $O(1)$*.*

We start our analysis of Algorithm 9.2 by stating that CORRECT-LABEL(S, δ) labels S correctly, with probability $1-\delta$. This is direct application of the Hoeffding bound and its proof is omitted.

Lemma 9.4.4. *For any unlabeled sample set* S*,* $\delta > 0$*, and* $\overline{S}$ = CORRECT-LABEL(S, δ)*, with probability* $1-\delta$*, for all* $(x, y) \in \overline{S}$*,* $y = f^*(x)$*.*

Note that as a direct consequence of the above lemma, Phase 1 of Algorithm 9.2 achieves error of $O(\sqrt{\epsilon})$.

Lemma 9.4.5. *In Algorithm 9.2, with probability* $1-\frac{\delta}{3}$*,* $\operatorname{err}_{\mathcal{D}}(h_1) \leq \frac{1}{2}\sqrt{\epsilon}$*.*

Next, we prove that FILTER removes instances that are correctly labeled by h_1 with good probability and retains instances that are mislabeled by h_1 with at least a constant probability.

Algorithm 9.2: BOOSTING BY PROBABILISTIC FILTERING FOR $\alpha = \frac{1}{2} + \Theta(1)$

Input: Given a distribution $\mathcal{D}_{|\mathcal{X}}$, a class of hypotheses $\mathcal{F}$, parameters ϵ and δ.
Phase 1
Let $\overline{S_1} = \text{CORRECT-LABEL}(S_1, \delta/6)$, for a set of sample S_1 of size $2m_{\sqrt{\epsilon},\delta/6}$ from $\mathcal{D}_{|\mathcal{X}}$.
Let $h_1 = \mathcal{O}_{\mathcal{F}}(\overline{S_1})$.

Phase 2
Let $S_I = \text{FILTER}(S_2, h_1)$, for a set of samples S_2 of size $\Theta(m_{\epsilon,\delta})$ drawn from $\mathcal{D}_{|\mathcal{X}}$.
Let S_C be a sample set of size $\Theta(m_{\sqrt{\epsilon},\delta})$ drawn from $\mathcal{D}_{|\mathcal{X}}$.
Let $\overline{S_{All}} = \text{CORRECT-LABEL}(S_I \cup S_C, \delta/6)$.
Let $\overline{W_I} = \{(x, y) \in \overline{S_{All}} \mid y \neq h_1(x)\}$ and Let $\overline{W_C} = \overline{S_{All}} \setminus \overline{W_I}$.
Draw a sample set $\overline{W}$ of size $\Theta(m_{\sqrt{\epsilon},\delta})$ from a distribution that equally weights $\overline{W_I}$ and $\overline{W_C}$.
Let $h_2 = \mathcal{O}_{\mathcal{F}}(\overline{W})$.

Phase 3
Let $\overline{S_3} = \text{CORRECT-LABEL}(S_3, \delta/6)$, for a sample set S_3 of size $2m_{\sqrt{\epsilon},\delta/6}$ drawn from $\mathcal{D}_{|\mathcal{X}}$ conditioned on $h_1(x) \neq h_2(x)$.
Let $h_3 = \mathcal{O}_{\mathcal{F}}(\overline{S_3})$.

return $\text{Maj}(h_1, h_2, h_3)$.

CORRECT-LABEL(S, δ)**:**

for $x \in S$ **do**
  Let $L \sim P^k$ for a set of $k = O(\log(\frac{|S|}{\delta}))$ labelers drawn from P
  $\overline{S} \leftarrow \overline{S} \cup \{(x, \text{Maj}_L(x))\}$.
end for
return $\overline{S}$.

Lemma 9.4.6. *Given any sample set S and classifier h, for every $x \in S$*

1. *If $h(x) = f^*(x)$, then $x \in \text{FILTER}(S, h)$ with probability $< \sqrt{\epsilon}$.*
2. *If $h(x) \neq f^*(x)$, then $x \in \text{FILTER}(S, h)$ with probability ≥ 0.5.*

Proof. For the first claim, note that $x \in S_I$ only if $\text{Maj}(y_{1:t}) \neq h(x)$ for all $t \leq N$. Consider $t = N$ time step. Since each random query agrees with $f^*(x) = h(x)$ with probability ≥ 0.7 independently, majority of $N = O(\log(1/\sqrt{\epsilon}))$ labels are correct with probability at least $1 - \sqrt{\epsilon}$. Therefore, the probability that the majority label disagrees with $h(x) = f^*(x)$ at every time step is at most $\sqrt{\epsilon}$.

In the second claim, we are interested in the probability that there exists some $t \leq N$, for which $\text{Maj}(y_{1:t}) = h(x) \neq f^*(x)$. This is the same as the probability of return in biased random walks, also called the probability of ruin in gambling [119], where we are given a random walk that takes a step to the right with probability ≥ 0.7 and takes a step to the left with the remaining probability and we are interested in the probability that this walk ever crosses the

origin to the left while taking N or even infinitely many steps. Using the probability of return for biased random walks (see Theorem C.0.1), the probability that $\mathrm{Maj}(y_{1:t}) \neq f^*(x)$ ever is at most $\left(1 - \left(\frac{0.7}{1-0.7}\right)^N\right) / \left(1 - \left(\frac{0.7}{1-0.7}\right)^{N+1}\right) < \frac{3}{7}$. Therefore, for each x such that $h(x) \neq f^*(x)$, $x \in S_I$ with probability at least $4/7$. □

In the remainder of the proof, for ease of exposition we assume that not only $\mathrm{err}_{\mathcal{D}}(h_1) \leq \frac{1}{2}\sqrt{\epsilon}$ as per Lemma 9.4.5, but in fact $\mathrm{err}_{\mathcal{D}}(h_1) = \frac{1}{2}\sqrt{\epsilon}$. This assumption is not needed for the correctness of the results but it helps simplify the notation and analysis. As a direct consequence of Lemma 9.4.6 and application of the Chernoff bound, we deduce that with high probability $\overline{W}_I$, $\overline{W}_C$, and S_I all have size $\Theta(m_{\sqrt{\epsilon},\delta})$.

Lemma 9.4.7. *With probability* $1 - \exp(-\Omega(m_{\sqrt{\epsilon},\delta}))$, $\overline{W}_I$, $\overline{W}_C$, *and* S_I *all have size* $\Theta(m_{\sqrt{\epsilon},\delta})$.

Proof. Let us first consider the expected size of sets S_I, $\overline{W}_I$, and $\overline{W}_C$. Using Lemma 9.4.6, we have

$$O(m_{\sqrt{\epsilon},\delta}) \geq \frac{1}{2}\sqrt{\epsilon}|S_2| + \sqrt{\epsilon}|S_2| \geq \mathbb{E}[|S_I|] \geq \frac{1}{2}\left(\frac{1}{2}\sqrt{\epsilon}\right)|S_2| \geq \Omega(m_{\sqrt{\epsilon},\delta}).$$

Similarly,

$$O(m_{\sqrt{\epsilon},\delta}) \geq \mathbb{E}[S_I] + |S_C| \geq \mathbb{E}[\overline{W}_I] \geq \frac{1}{2}\left(\frac{1}{2}\sqrt{\epsilon}\right)|S_2| \geq \Omega(m_{\sqrt{\epsilon},\delta}).$$

Similarly,

$$O(m_{\sqrt{\epsilon},\delta}) \geq \mathbb{E}[S_I] + |S_C| \geq \mathbb{E}[\overline{W}_C] \geq \left(1 - \frac{1}{2}\sqrt{\epsilon}\right)|S_C| \geq \Omega(m_{\sqrt{\epsilon},\delta}).$$

The claim follows by the Chernoff bound. □

The next lemma combines the probabilistic filtering and super-sampling techniques to show that h_2 has the desired error $O(\sqrt{\epsilon})$ on $\mathcal{D}_2$.

Lemma 9.4.8. *Let* $\mathcal{D}_C$ *and* $\mathcal{D}_I$ *denote distribution* $\mathcal{D}$ *when it is conditioned on* $\{x \mid h_1(x) = f^*(x)\}$ *and* $\{x \mid h_1(x) \neq f^*(x)\}$, *respectively, and let* $\mathcal{D}_2 = \frac{1}{2}\mathcal{D}_I + \frac{1}{2}\mathcal{D}_C$. *With probability* $1 - 2\delta/3$, $\mathrm{err}_{\mathcal{D}_2}(h_2) \leq \frac{1}{2}\sqrt{\epsilon}$.

Proof. Consider distribution $\mathcal{D}'$ that has equal probability on the distributions induced by $\overline{W_I}$ and $\overline{W_C}$ and let $d'(x)$ denote the density of point x in this distribution. Relying on our *super-sampling* technique (see Lemma 9.4.2), it is sufficient to show that for any x, $d'(x) = \Theta(d_2(x))$.

For ease of presentation, we assume that Lemma 9.4.5 holds with equality, i.e., $\mathrm{err}_{\mathcal{D}}(h_1)$ is exactly $\frac{1}{2}\sqrt{\epsilon}$ with probability $1 - \delta/3$. Let $d(x)$, $d_2(x)$, $d_C(x)$, and $d_I(x)$ be the density of instance x in distributions $\mathcal{D}$, $\mathcal{D}_2$, $\mathcal{D}_C$, and $\mathcal{D}_I$, respectively. Note that, for any x such that $h_1(x) = f^*(x)$, we have $d(x) = d_C(x)(1 - \frac{1}{2}\sqrt{\epsilon})$. Similarly, for any x such that $h_1(x) \neq f^*(x)$, we have $d(x) = d_I(x)\frac{1}{2}\sqrt{\epsilon}$. Let $N_C(x)$, $N_I(x)$, $M_C(x)$ and $M_I(x)$ be the number of occurrences of x in the sets S_C, S_I, $\overline{W_C}$ and $\overline{W_I}$, respectively. For any x, there are two cases:

If $h_1(x) = f^*(x)$: Then, there exist absolute constants c_1 and c_2 according to Lemma 9.4.7, such that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_C(x)}{|\overline{W_C}|}\right] \geq \frac{\mathbb{E}[M_C(x)]}{c_1 \cdot m_{\sqrt{\epsilon},\delta}} \geq \frac{\mathbb{E}[N_C(x)]}{c_1 \cdot m_{\sqrt{\epsilon},\delta}} = \frac{|S_C| \cdot d(x)}{c_1 \cdot m_{\sqrt{\epsilon},\delta}}$$
$$= \frac{|S_C| \cdot d_C(x) \cdot (1 - \frac{1}{2}\sqrt{\epsilon})}{c_1 \cdot m_{\sqrt{\epsilon},\delta}} \geq c_2 d_C(x) = \frac{c_2 d_2(x)}{2},$$

where the second and sixth transitions are by the sizes of $\overline{W_C}$ and $|S_C|$ and the third transition is by the fact that if $h(x) = f^*(x)$, $M_C(x) > N_C(x)$.
If $h_1(x) \neq f^*(x)$: Then, there exist absolute constants c_1' and c_2' according to Lemma 9.4.7, such that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_I(x)}{|\overline{W_I}|}\right] \geq \frac{\mathbb{E}[M_I(x)]}{c_1' \cdot m_{\sqrt{\epsilon},\delta}} \geq \frac{\mathbb{E}[N_I(x)]}{c_1' \cdot m_{\sqrt{\epsilon},\delta}} \geq \frac{\frac{4}{7}\,d(x)|S_2|}{c_1' \cdot m_{\sqrt{\epsilon},\delta}}$$
$$= \frac{\frac{4}{7}\,d_I(x)\frac{1}{2}\sqrt{\epsilon} \cdot |S_2|}{c_1' \cdot m_{\sqrt{\epsilon},\delta}} \geq c_2' d_I(x) = \frac{c_2' d_2(x)}{2},$$

where the second and sixth transitions are by the sizes of $\overline{W_I}$ and $|S_2|$, the third transition is by the fact that if $h(x) \neq f^*(x)$, $M_I(x) > N_I(x)$, and the fourth transition holds by part 2 of Lemma 9.4.6.

Using the super-sampling guarantees of Lemma 9.4.2, with probability $1-2\delta/3$, $\mathrm{err}_{\mathcal{D}_2}(h_2) \leq \sqrt{\epsilon}/2$. □

The next claim shows that the probabilistic filtering step queries a few labels only. At a high level, this is achieved by showing that any instance x for which $h_1(x) = f^*(x)$ contributes only $O(1)$ queries, with high probability. On the other hand, instances that h_1 mislabeled may each get $\log(\frac{1}{\epsilon})$ queries. But, because there are only few such points, the total number of queries these instances require is a lower order term.

Lemma 9.4.9. *Let S be a sample set drawn from distribution $\mathcal{D}$ and let h be such that $\mathrm{err}_{\mathcal{D}}(h) \leq \sqrt{\epsilon}$. With probability $1 - \exp(-\Omega(|S|\sqrt{\epsilon}))$, $\textsc{Filter}(S, h)$ makes $O(|S|)$ label queries.*

Proof. Using Chernoff bound, with probability $1 - \exp\left(-|S|\sqrt{\epsilon}\right)$ the total number of points in S where h disagrees with f^* is $O(|S|\sqrt{\epsilon})$. The number of queries spent on these points is at most $O\left(|S|\sqrt{\epsilon}\log(1/\epsilon)\right) \leq O(|S|)$.

Next, we show that for each x such that $h(x) = f^*(x)$, the number of queries taken until a majority of them agree with $h(x)$ is a constant. Let us first show that this is the case in expectation. Let N_i be the expected number of labels queried until we have i more correct labels than incorrect ones. Then $N_1 \leq 0.7(1) + 0.3(N_2 + 1)$, since with probability at least $\alpha \geq 0.7$, we receive one more correct label and stop, and with probability ≤ 0.3 we get a wrong label in which case we have to get two more correct labels in future. Moreover, $N_2 = 2N_1$, since we have to get one more correct label to move from N_2 to N_1 and then one more. Solving these, we have that $N_1 \leq 2.5$. Therefore, the expected total number of queries is at most $O(|S|)$. Next, we show that this random variable is also well-concentrated. Let L_x be a random variable that

indicates the total number of queries on x before we have one more correct label than incorrect labels. Note that L_x is an unbounded random variable, therefore concentration bounds such as Hoeffding or Chernoff do not work here. Instead, to show that L_x is well-concentrated, we prove that the Bernstein inequality (see Appendix C) holds.

We prove that the Bernstein inequality holds for the total number of queries $y_1, y_2, \ldots,$ made before their majority agrees with $f^*(x)$. Let L_x be the random variable denoting the number of queries the algorithm makes on instance x for which $h(x) = f^*(x)$. Consider the probability that $L_x = 2k+1$ for some k. That is, $\mathrm{Maj}(y_{1:t}) = f^*(x)$ for the first time when $t = 2k+1$. This is at most the probability that $\mathrm{Maj}(y_{1:2k-1}) \neq f^*(x)$. By Chernoff bound, we have that

$$\begin{aligned}\Pr[L_x = 2k+1] &\leq \Pr[\mathrm{Maj}(y_{1:2k-1}) \neq f^*(x)] \leq \exp\left(-0.7(2k-1)(\frac{2}{7})^2/2\right) \\ &\leq \exp\left(-0.02(2k-1)\right).\end{aligned}$$

For each $i > 1$, we have

$$\begin{aligned}\mathbb{E}[(L_x - \mathbb{E}[L_x])^i] &\leq \sum_{k=0}^{\infty} \Pr[L_x = 2k+1](2k+1-\mathbb{E}[L_x])^i \\ &\leq \sum_{k=0}^{\infty} e^{-0.02(2k-1)}(2k+1)^i \\ &\leq e^{0.04}\sum_{k=0}^{\infty} e^{-0.02(2k+1)}(2k+1)^i \\ &\leq e^{0.04}\sum_{k=0}^{\infty} e^{-0.02k}k^i \\ &\leq 50(i+1)!\, e^{4i+0.04},\end{aligned}$$

where the last inequality is done by integration. This satisfies the Bernstein condition (See Appendix C). Therefore,

$$\Pr\left[\sum_{x\in S} L_x - |S|\,\mathbb{E}[L_x] \geq O(|S|)\right] \leq \exp\left(-|S|\right).$$

Therefore, the total number of queries over all points in $x \in S$ where $h(x) = f^*(x)$ is at most $O(|S|)$ with very high probability. □

Finally, we have all of the ingredients needed for proving our main theorem.

Proof of Theorem 9.4.3. We first discuss the number of label queries Algorithm 9.2 makes. The total number of labels queried in Phases 1 and 3 is attributed to CORRECT-LABEL(S_1, δ) and CORRECT-LABEL$(S_3, \delta/6)$, which is $O\left(m_{\sqrt{\epsilon},\delta}\log(m_{\sqrt{\epsilon},\delta}/\delta)\right)$. By Lemma 9.4.7, $|S_I \cup S_C| \leq O(m_{\sqrt{\epsilon},\delta})$ almost surely. So CORRECT-LABEL$(S_I \cup S_C, \delta/6)$ takes $O\left(m_{\sqrt{\epsilon},\delta}\log(m_{\sqrt{\epsilon},\delta}/\delta)\right)$ labels. Moreover, as we showed in Lemma 9.4.9, FILTER(S_2, h_1) queries $O(m_{\epsilon,\delta})$ labels, almost

surely. So, the total number of labels queried by Algorithm 9.2 is $O\left(m_{\sqrt{\epsilon},\delta}\log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right)+m_{\epsilon,\delta}\right)$. This leads to $\Lambda = O\left(\sqrt{\epsilon}\log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right)+1\right)$ cost per labeled example.

It remains to show that $MAJ(h_1,h_2,h_3)$ has error $\leq \epsilon$ on $\mathcal{D}$. Since CORRECT-LABEL$(S_1,\delta/6)$ and CORRECT-LABEL$(S_3,\delta/6)$ return correctly labeled sets , $\mathrm{err}_{\mathcal{D}}(h_1)\leq\frac{1}{2}\sqrt{\epsilon}$ and $\mathrm{err}_{\mathcal{D}_3}(h_3)\leq\frac{1}{2}\sqrt{\epsilon}$, where $\mathcal{D}_3$ is distribution $\mathcal{D}$ conditioned on $\{x \mid h_1(x)\neq h_2(x)\}$. As we showed in Lemma 9.4.8, $\mathrm{err}_{\mathcal{D}_2}(h_2)\leq\frac{1}{2}\sqrt{\epsilon}$ with probability $1-2\delta/3$. Using the boosting technique of Schapire [243] described in Theorem 9.4.1, we conclude that $MAJ(h_1,h_2,h_3)$ has error $\leq\epsilon$ on $\mathcal{D}$. □

9.4.1 The General Case of Any α

In this section, we extend Algorithm 9.2 to handle any value of α, that does not necessarily satisfy $\alpha > \frac{1}{2}+\Theta(1)$. We show that by using $O(\frac{1}{\alpha})$ golden queries, it is possible to efficiently learn any function class with a small overhead.

There are two key challenges that one needs to overcome when $\alpha < \frac{1}{2}+o(1)$. First, we can no longer assume that by taking the majority vote over a few random labelers we get the correct label of an instance. Therefore, CORRECT-LABEL(S,δ) may return a highly noisy labeled sample set. This is problematic, since efficiently learning h_1,h_2, and h_3 using oracle $\mathcal{O}_{\mathcal{F}}$ crucially depends on the correctness of the input labeled set. Second, FILTER(S,h_1) no longer "filters" the instances correctly based on the classification error of h_1. In particular, FILTER may retain a constant fraction of instances where h_1 is in fact correct, and it may throw out instances where h_1 was incorrect with high probability. Therefore, the per-instance guarantees of Lemma 9.4.6 fall apart, immediately.

We overcome both of these challenges by using two key ideas outlined below.

Pruning: As we alluded to in Section 9.3, instances where only a small majority of labelers are in agreement are great for identifying and pruning away a noticeable fraction of the bad labelers. We call these instances *good test cases*. In particular, if we ever encounter a good test case x, we can ask a golden query $y=f^*(x)$ and from then on only consider the labelers who got this test correctly, i.e., $P\leftarrow P_{|\{(x,y)\}}$. Note that if we make our golden queries when $\text{Maj-size}_P(x)\leq 1-\frac{\alpha}{2}$, at least an $\frac{\alpha}{2}$ fraction of the labelers would be pruned. This can be repeated at most $O(\frac{1}{\alpha})$ times before the number of good labelers form a strong majority, in which case Algorithm 9.2 succeeds. The natural question is how would we measure $\text{Maj-size}_P(x)$ using few label queries? Interestingly, CORRECT-LABEL(S,δ) can be modified to detect such good test cases by measuring the empirical agreement rate on a set L of $O(\frac{1}{\alpha^2}\log(\frac{|S|}{\delta}))$ labelers. This is shown in procedure PRUNE-AND-LABEL as part Algorithm 9.3. That is, if $\text{Maj-size}_L(x) > 1-\alpha/4$, we take $\text{Maj}_L(x)$ to be the label, otherwise we test and prune the labelers, and then restart the procedure. This ensures that whenever we use a sample set that is labeled by PRUNE-AND-LABEL, we can be certain of the correctness of the labels.

Lemma 9.4.10. *For any unlabeled sample set S, $\delta>0$, with probability $1-\delta$, we have that either* PRUNE-AND-LABEL(S,δ) *prunes the set of labelers or* $\overline{S}=$ PRUNE-AND-LABEL(S,δ) *is such that for all $(x,y)\in\overline{S}$, $y=f^*(x)$.*

Proof. By Chernoff bound, with probability $\geq 1-\delta$, for every $x \in S$ we have that

$$|\text{Maj-size}_P(x) - \text{Maj-size}_L(x)| \leq \frac{\alpha}{8},$$

where L is the set of labelers PRUNE-AND-LABEL(S, δ) queries on x. Hence, if x is such that $\text{Maj-size}_P(x) \leq 1 - \frac{\alpha}{2}$, then it will be identified and the set of labelers is pruned. Otherwise, $\text{Maj}_L(x)$ agrees with the good labelers and x gets labeled correctly according to the target function. □

As an immediate result, the first phase of Algorithm 9.3 succeeds in computing h_1, such that $\text{err}_{\mathcal{D}}(h_1) \leq \frac{1}{2}\sqrt{\epsilon}$. Moreover, every time PRUNE-AND-LABEL prunes the set of labelers, the total fraction of good labeler among all remaining labelers increase. As we show, after $O(1/\alpha)$ prunings, the set of good labelers is guaranteed to form a large majority, in which case Algorithm 9.2 for the case of $\alpha = \frac{1}{2} + \Theta(1)$ can be used. This is stated in the next lemma.

Lemma 9.4.11. *For any δ, with probability $1-\delta$, the total number of times that Algorithm 9.3 is restarted as a result of pruning is $O(\frac{1}{\alpha})$.*

Proof. Recall that $\delta' = c \cdot \alpha\delta$ for some small enough constant $c > 0$. For every call to PRUNE-AND-LABEL(S, δ'), by Hoeffding bound, it is guaranteed that with probability $\geq 1-\delta'$, for each $x \in S$,

$$|\text{Maj-size}_P(x) - \text{Maj-size}_L(x)| \leq \frac{\alpha}{8},$$

where L is the set of labelers PRUNE-AND-LABEL(S, δ') queries on x. Hence, when we issue a golden query for x such that $\text{Maj-size}_L(x) \leq 1 - \frac{\alpha}{4}$ and prune away bad labelers, we are guaranteed to remove at least an $\frac{\alpha}{8}$ fraction of the labelers. Furthermore, no good labeler is ever removed. Hence, the fraction of good labelers increases from α to $\alpha/(1-\frac{\alpha}{8})$. So, in $O(\frac{1}{\alpha})$ calls, the fraction of the good labelers surpasses $\frac{3}{4}$ and we switch to using Algorithm 9.2. Therefore, with probability $1-\delta$ overall, the total number of golden queries is $O(1/\alpha)$.

□

Robust Super-sampling: The filtering step faces a completely different challenge: Any point that is a good test case can be filtered the wrong way. However, instances where still a strong majority of the labelers agree are not affected by this problem and will be filtered correctly. Therefore, as a first step we ensure that the total number of good test cases that were not caught before FILTER starts is small. For this purpose, we start the algorithm by calling CORRECT-LABEL on a sample of size $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$, and if no test points were found in this set, then with high probability the total fraction of good test cases in the underlying distribution is at most $\frac{\epsilon}{2}$. Since the fraction of good test cases is very small, one can show that except for an $\sqrt{\epsilon}$ fraction, the noisy distribution constructed by the filtering process will, for the purposes of boosting, satisfy the conditions needed for the super-sampling technique. Here, we introduce a robust version of the super-sampling technique to argue that the filtering step will indeed produce h_2 of error $O(\sqrt{\epsilon})$.

Lemma 9.4.12 (Robust Super-Sampling Lemma)**.** *Given a hypothesis class $\mathcal{F}$ consider any two discrete distributions $\mathcal{D}$ and $\mathcal{D}'$ such that except for an ϵ fraction of the mass under $\mathcal{D}$, we have*

that for all x, $d'(x) \geq c \cdot d(x)$ for an absolute constant $c > 0$ and both distributions are labeled according to $f^* \in \mathcal{F}$. There exists a constant $c' > 1$ such that for any ϵ and δ, with probability $1 - \delta$ over a labeled sample set S of size $c' m_{\epsilon,\delta}$ drawn from $\mathcal{D}'$, $\mathcal{O}_{\mathcal{F}}(S)$ has error of at most 2ϵ with respect to $\mathcal{D}$.

Proof. Let B be the set of points that do not satisfy the condition that $d'(x) \geq c \cdot d(x)$. Notice that because $\mathcal{D}$ and $\mathcal{D}'$ are both labeled according to $f^* \in \mathcal{F}$, for any $f \in \mathcal{F}$ we have,

$$\operatorname*{err}_{\mathcal{D}'}(f) = \sum_{x \in B} d'(x) \mathbb{1}_{f(x) \neq f^*(x)} + \sum_{x \notin B} d'(x) \mathbb{1}_{f(x) \neq f^*(x)} \geq \sum_{x \notin B} c \cdot d(x) \mathbb{1}_{f(x) \neq f^*(x)} \geq c \cdot (\operatorname*{err}_{\mathcal{D}}(f) - \epsilon).$$

Therefore, if $\operatorname{err}_{\mathcal{D}'}(f) \leq c\epsilon$, then $\operatorname{err}_{\mathcal{D}}(f) \leq 2\epsilon$. Let $m' = m_{c\epsilon,\delta}$, we have

$$\begin{aligned} \delta &> \Pr_{S' \sim \mathcal{D}'^{m'}}[\exists f \in \mathcal{F}, \text{s.t. } \operatorname*{err}_{S'}(f) = 0 \wedge \operatorname*{err}_{\mathcal{D}'}(f) \geq c\epsilon] \\ &\geq \Pr_{S' \sim \mathcal{D}'^{m'}}[\exists f \in \mathcal{F}, \text{s.t. } \operatorname*{err}_{S'}(f) = 0 \wedge \operatorname*{err}_{\mathcal{D}}(f) \geq 2\epsilon]. \end{aligned}$$

The claim follows by the fact that $m_{c\epsilon,\delta} = O\left(\frac{1}{c} m_{\epsilon,\delta}\right)$.

□

By combining these techniques at every execution of our algorithm we ensure that if a good test case is ever detected we prune a small fraction of the bad labelers and restart the algorithm, and if it is never detected, our algorithm returns a classifier of error ϵ.

Theorem 9.4.13 (Any α). *Suppose the fraction of the perfect labelers is α and let $\delta' = c\alpha\delta$ for small enough constant $c > 0$. Algorithm 9.3 uses oracle $\mathcal{O}_{\mathcal{F}}$, runs in time $\operatorname{poly}(d, \frac{1}{\alpha}, \frac{1}{\epsilon}, \ln(\frac{1}{\delta}))$, uses a training set of size $O(\frac{1}{\alpha} m_{\epsilon,\delta'})$ size and with probability $1 - \delta$ returns $f \in \mathcal{F}$ with $\operatorname{err}_{\mathcal{D}}(f) \leq \epsilon$ using $O(\frac{1}{\alpha})$ golden queries, load of $\frac{1}{\alpha}$ per labeler, and a total number of queries*

$$O\left(\frac{1}{\alpha} m_{\epsilon,\delta'} + \frac{1}{\alpha\epsilon} \log(\frac{1}{\delta'}) \log(\frac{1}{\epsilon\delta'}) + \frac{1}{\alpha^3} m_{\sqrt{\epsilon},\delta'} \log(\frac{m_{\sqrt{\epsilon},\delta'}}{\delta'})\right).$$

Note that when $\frac{1}{\alpha^2\sqrt{\epsilon}} \geq \log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\alpha\delta}\right)$ and $\log(\frac{1}{\alpha\delta}) < d$, the cost per labeled query is $O(\frac{1}{\alpha})$.

Proof. Recall that $\delta' = c \cdot \alpha\delta$ for a small enough constant $c > 0$. Let $B = \{x \mid \text{Maj-size}_P(x) \leq 1 - \alpha/2\}$ be the set of good test cases and and let $\beta = D[B]$ be the total density on such points. Note that if $\beta > \frac{\epsilon}{4}$, with high probability S_0 includes one such point, in which case PRUNE-AND-LABEL identifies it and prunes the set of labelers. Therefore, we can assume that $\beta \leq \frac{\epsilon}{4}$. By Lemma 9.4.10, it is easy to see that $err_{\mathcal{D}}(h_1) \leq \frac{1}{2}\sqrt{\epsilon}$.

We now analyze the filtering step of Phase 2. As in Section 9.4, our goal is to argue that $err_{\mathcal{D}_2}(h_2) \leq \frac{1}{2}\sqrt{\epsilon}$. Consider distribution $\mathcal{D}'$ that has equal probability on the distributions induced by $\overline{W_I}$ and $\overline{W_C}$ and let $d'(x)$ denote the density of point x in this distribution. We will show that for any $x \notin B$ we have that $d'(x) = \Theta(d_2(x))$. Since $\mathcal{D}[B] \leq \frac{\epsilon}{4}$, we have that $\mathcal{D}_2[B] \leq \frac{1}{4}\sqrt{\epsilon}$. Therefore, $\mathcal{D}'$ and $\mathcal{D}_2$ satisfy the conditions of the robust super-sampling lemma (Lemma 9.4.12) where the fraction of bad points is at most $\frac{\sqrt{\epsilon}}{4}$. Hence, $err_{\mathcal{D}_2}(h_2) \leq \frac{1}{2}\sqrt{\epsilon}$.

Algorithm 9.3: BOOSTING BY PROBABILISTIC FILTERING FOR ANY α

Input: Given a distribution $\mathcal{D}_{|\mathcal{X}}$ and P, a class of hypothesis $\mathcal{F}$, parameters ϵ, δ, and α.

Phase 0

If $\alpha > \frac{3}{4}$, run Algorithm 9.2 and quit.

Let $\delta' = c\alpha\delta$ for small enough $c > 0$ and draw S_0 of $O(\frac{1}{\epsilon}\log(\frac{1}{\delta'}))$ examples from $\mathcal{D}$.

PRUNE-AND-LABEL(S_0, δ').

Phase 1

Let $\overline{S_1}$ = PRUNE-AND-LABEL(S_1, δ'), for a set of sample S_1 of size $2m_{\sqrt{\epsilon},\delta'}$ from $\mathcal{D}$.

Let $h_1 = \mathcal{O}_{\mathcal{F}}(\overline{S_1})$.

Phase 2

Let S_I = FILTER(S_2, h_1), for a set of samples S_2 of size $\Theta(m_{\epsilon,\delta'})$ drawn from $\mathcal{D}$.

Let S_C be a sample set of size $\Theta(m_{\sqrt{\epsilon},\delta'})$ drawn from $\mathcal{D}$.

Let $\overline{S_{All}}$ = PRUNE-AND-LABEL$(S_I \cup S_C, \delta')$.

Let $\overline{W_I} = \{(x, y) \in \overline{S_{All}} \mid y \neq h_1(x)\}$ and Let $\overline{W_C} = \overline{S_{All}} \setminus \overline{W_I}$.

Draw a sample set $\overline{W}$ of size $\Theta(m_{\sqrt{\epsilon},\delta'})$ from a distribution that equally weights $\overline{W_I}$ and $\overline{W_C}$.

Let $h_2 = \mathcal{O}_{\mathcal{F}}(\overline{W})$.

Phase 3

Let $\overline{S_3}$ = PRUNE-AND-LABEL(S_3, δ'), for a sample set S_3 of size $2m_{\sqrt{\epsilon},\delta'}$ drawn from $\mathcal{D}$ conditioned on $h_1(x) \neq h_2(x)$.

Let $h_3 = \mathcal{O}_{\mathcal{F}}(\overline{S_3})$.

return $\mathrm{Maj}(h_1, h_2, h_3)$.

Algorithm 9.4: PRUNE-AND-LABEL(S, δ)

for $x \in S$ **do**
  Let $L \sim P^k$ for a set of $k = O(\frac{1}{\alpha^2}\log(\frac{|S|}{\delta}))$ labelers drawn from P.
  if $\text{Maj-size}_L(x) \leq 1 - \frac{\alpha}{4}$ **then**
    Get a golden query $y^* = f^*(x)$.
    Restart Algorithm 9.3 with distribution $P \leftarrow P_{|\{(x,y^*)\}}$ and $\alpha \leftarrow \frac{\alpha}{1-\frac{\alpha}{8}}$.
  else
    $\overline{S} \leftarrow \overline{S} \cup \{(x, \mathrm{Maj}_L(x))\}$
  end if
end for
return $\overline{S}$

We now show that for any $x \in B$, $d'(x) = \Theta(d_2(x))$. The proof is identical to the one in Lemma 9.4.8. For ease of representation, we assume that $\text{err}_{\mathcal{D}}(h_1)$ is exactly $\frac{1}{2}\sqrt{\epsilon}$. Let $d(x)$, $d_2(x)$, $d_C(x)$, and $d_I(x)$ be the density of instance x in distributions $\mathcal{D}$, $\mathcal{D}_2$, $\mathcal{D}_C$, and $\mathcal{D}_I$, respectively. Note that, for any x such that $h_1(x) = f^*(x)$, we have $d(x) = d_C(x)(1 - \frac{1}{2}\sqrt{\epsilon})$. Similarly, for any x such that $h_1(x) \neq f^*(x)$, we have $d(x) = d_I(x)\frac{1}{2}\sqrt{\epsilon}$. Let $N_C(x)$, $N_I(x)$, $M_C(x)$ and $M_I(x)$ be the number of occurrences of x in the sets S_C, S_I, $\overline{W_C}$ and $\overline{W_I}$, respectively. For any x, there are two cases:

If $h_1(x) = f^*(x)$: Then, there exist absolute constants c_1 and c_2 according to Lemma 9.4.7, such that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_C(x)}{|\overline{W_C}|}\right] \geq \frac{\mathbb{E}[M_C(x)]}{c_1 \cdot m_{\sqrt{\epsilon},\delta}} \geq \frac{\mathbb{E}[N_C(x)]}{c_1 \cdot m_{\sqrt{\epsilon},\delta}} = \frac{|S_C| \cdot d(x)}{c_1 \cdot m_{\sqrt{\epsilon},\delta}}$$
$$= \frac{|S_C| \cdot d_C(x) \cdot (1 - \frac{1}{2}\sqrt{\epsilon})}{c_1 \cdot m_{\sqrt{\epsilon},\delta}} \geq c_2 d_C(x) = \frac{c_2 d_2(x)}{2},$$

where the second and sixth transitions are by the sizes of $\overline{W_C}$ and $|S_C|$ and the third transition is by the fact that if $h(x) = f^*(x)$, $M_C(x) > N_C(x)$.

If $h_1(x) \neq f^*(x)$: Then, there exist absolute constants c_1' and c_2' according to Lemma 9.4.7, such that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_I(x)}{|\overline{W_I}|}\right] \geq \frac{\mathbb{E}[M_I(x)]}{c_1' \cdot m_{\sqrt{\epsilon},\delta}} \geq \frac{\mathbb{E}[N_I(x)]}{c_1' \cdot m_{\sqrt{\epsilon},\delta}} \geq \frac{0.5\; d(x)|S_2|}{c_1' \cdot m_{\sqrt{\epsilon},\delta}}$$
$$= \frac{0.5\; d_I(x)\frac{1}{2}\sqrt{\epsilon} \cdot |S_2|}{c_1' \cdot m_{\sqrt{\epsilon},\delta}} = c_2' d_I(x) = \frac{c_2' d_2(x)}{2},$$

where the second and sixth transitions are by the sizes of $\overline{W_I}$ and $|S_2|$, the third transition is by the fact that if $h(x) \neq f^*(x)$, $M_I(x) > N_I(x)$, and the fourth transition holds by part 2 of Lemma 9.4.6.

Finally, we have that $\text{err}_{\mathcal{D}_3}(h_3) \leq \frac{1}{2}\sqrt{\epsilon}$, where $\mathcal{D}_3$ is distribution $\mathcal{D}$ conditioned on $\{x \mid h_1(x) \neq h_2(x)\}$. Using the boosting technique of Schapire [243] describe in Theorem 9.4.1, we conclude that $MAJ(h_1, h_2, h_3)$ has error $\leq \epsilon$ on $\mathcal{D}$.

The label complexity claim follows by the fact that we restart Algorithm 9.3 at most $O(1/\alpha)$ times, take an additional $O(\frac{1}{\epsilon}\log(\frac{1}{\delta'}))$ high quality labeled set, and each run of Algorithm 9.3 uses the same label complexity as in Theorem 9.4.3 before getting restarted. □

9.5 No Perfect Labelers

In this section, we consider a scenario where our pool of labelers does not include any perfect labelers. Unfortunately, learning f^* in this setting reduces to the notoriously difficult agnostic learning problem. A related task is to find a set of the labelers which are *good* but not perfect. In this section, we show how to identify the set of all good labelers, when at least the majority of the labelers are good.

We consider a setting where the fraction of the perfect labelers, α, is arbitrarily small or 0. We further assume that at least half of the labelers are good, while others have considerably worst performance. More formally, we are given a set of labelers $g_1, \ldots, g_n$ and a distribution $\mathcal{D}$ with an unknown target classifier $f^* \in \mathcal{F}$. We assume that more than half of these labelers are "good", that is they have error of $\leq \epsilon$ on distribution $\mathcal{D}$. On the other hand, the remaining labelers, which we call "bad", have error rates $\geq 4\epsilon$ on distribution $\mathcal{D}$. We are interested in identifying all of the good labelers with high probability by querying the labelers on an unlabeled sample set drawn from $\mathcal{D}_{|\mathcal{X}}$.

This model presents an interesting community structure: Two good labelers agree on at least $1 - 2\epsilon$ fraction of the data, while a bad and a good labeler agree on at most $1 - 3\epsilon$ of the data. Note that the rate of agreement between two bad labelers can be arbitrary. This is due to the fact that there can be multiple bad labelers with the same classification function, in which case they completely agree with each other, or two bad labelers who disagree on the classification of every instance. This structure serves as the basis of Algorithm 9.5 and its analysis. Here we provide an overview of its working and analysis.

Algorithm 9.5: GOOD LABELER DETECTION

1: Input: Given n labelers, parameters ϵ and δ
2: Let $G = ([n], \emptyset)$ be a graph on n vertices with no edges.
3: Take set Q of $16\ln(2)n$ random pairs of nodes from G.
4: **for** $(i, j) \in Q$ **do**
5: **if** DISAGREE$(i, j) < 2.5\epsilon$ **then** add edge (i, j) to G **end if**
6: **end for**
7: Let $\mathcal{C}$ be the set of connected components of G each with $\geq n/4$ nodes.
8: **for** $i \in [n] \setminus \left(\bigcup_{C \in \mathcal{C}} C\right)$ and $C \in \mathcal{C}$ **do**
9: Take one node $j \in C$, if DISAGREE$(i, j) < 2.5\epsilon$ add edge (i, j) to G.
10: **end for**
11: **return** The largest connected component of G

12: **DISAGREE**(i, j)**:**

13: Take set S of $\Theta(\frac{1}{\epsilon}\ln(\frac{n}{\delta}))$ samples from $\mathcal{D}$.
14: **return** $\frac{1}{|S|}\sum_{x \in S} \mathbb{1}_{(g_i(x) \neq g_j(x))}$.

Theorem 9.5.1 (Informal)**.** *Suppose that any* good *labeler i is such that $\mathrm{err}_{\mathcal{D}}(g_i) \leq \epsilon$. Furthermore, assume that $\mathrm{err}_{\mathcal{D}}(g_j) \notin (\epsilon, 4\epsilon)$ for any $j \in [n]$. And let the number of good labelers be at least $\lfloor \frac{n}{2} \rfloor + 1$. Then, Algorithm 9.5, returns the set of all good labeler with probability $1 - \delta$, using an expected load of $\lambda = O\left(\frac{1}{\epsilon}\ln\left(\frac{n}{\delta}\right)\right)$ per labeler.*

We view the labelers as nodes in a graph that has no edges at the start of the algorithm. In step 4, the algorithm takes $O(n)$ random pairs of labelers and estimates their level of disagreement by querying them on an unlabeled sample set of size $O\left(\frac{1}{\epsilon}\ln\left(\frac{n}{\delta}\right)\right)$ and measuring their empirical disagreement. By an application of Chernoff bound, we know that with probability $1 - \delta$, for any $i, j \in [n]$,

$$\left|\textsc{Disagree}(i, j) - \Pr_{x \sim \mathcal{D}_{|\mathcal{X}}}[g_i(x) \neq g_j(x)]\right| < \frac{\epsilon}{2}.$$

Therefore, for any pair of good labelers i and j tested by the algorithm, $\textsc{Disagree}(i,j) < 2.5\epsilon$, and for any pair of labelers i and j that one is good and the other is bad, $\textsc{Disagree}(i,j) \geq 2.5\epsilon$. Therefore, the connected components of such a graph only include labelers from a single community.

Next, we show that at step 7 of Algorithm 9.5 with probability $1-\delta$ there exists at least one connected component of size $n/4$ of good labelers.

To see this we first prove that for any two good labelers i and j, the probability of (i,j) existing is at least $\Theta(1/n)$. Let V_g be the set of nodes corresponding to good labelers. For $i,j \in V_g$, we have

$$\Pr[(i,j) \in G] = 1 - \left(1 - \frac{1}{n^2}\right)^{4\ln(2)n} \approx \frac{4\ln(2)}{n} \geq \frac{2\ln(2)}{|V_g|}.$$

By the properties of random graphs, with very high probability there is a component of size $\beta|V_g|$ in a random graph whose edges exists with probability $c/|V_g|$, for $\beta + e^{-\beta c} = 1$ [165]. Therefore, with probability $1-\delta$, there is a component of size $|V_g|/2 > n/4$ over the vertices in V_g.

Finally, at step 8 the algorithm considers smaller connected components and tests whether they join any of the bigger components, by measuring the disagreement of two arbitrary labelers from these components.,At this point, all good labelers form one single connected component of size $> \frac{n}{2}$. So, the algorithm succeeds in identifying all good labelers.

Next, we briefly discuss the expected load per labeler in Algorithm 9.5. Each labeler participates in $O(1)$ pairs of disagreement tests in expectation, each requiring $O(\frac{1}{\epsilon}\ln(n/\delta))$ queries. So, in expectation each labeler labels $O(\frac{1}{\epsilon}\ln(n/\delta))$ instances.

Part III

Learning by People

Chapter 10

Collaborative PAC Learning

10.1 Introduction

With the wide application of machine learning methods to many aspects of day-to-day life, many simultaneous learning processes may be analyzing the same or related concepts at any given moment. In this chapter, we consider collaborative learning, where two or more learners attempt to learn together, capitalizing on one another's resources and by asking one another for information. Indeed, it is self-evident that collaboration, and the sharing of information, can make learning more efficient. In this chapter, we formalize and quantify this intuition and study its implications.

As an example, suppose k branches of a department store, which have sales data for different items in different locations, wish to collaborate on learning which items should be sold at each location. In this case, we would like to use the sales information across different branches to learn a good policy for each branch. Another example is given by k hospitals with different patient demographics, e.g., in terms of racial or socio-economic factors, which want to predict occurrence of a disease in patients. In addition to requiring a classifier that performs well on the population served by each hospital, it is natural to assume that all hospitals deploy a common classifier.

Motivated by these examples, we consider a model of *collaborative PAC learning*, in which k players attempt to learn the same underlying concept. We then ask how much information is needed for all players to simultaneously succeed in learning a desirable classifier. Specifically, we focus on the classic *probably approximately correct (PAC)* setting of Valiant [262], where there is an unknown target function $h^* \in \mathcal{H}$. We consider k players with distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$ that are labeled according to h^*. Our goal is to learn h^* up to an error of ϵ on *each and every* player distribution while requiring only a small number of samples overall.

A natural but naïve algorithm that forgoes collaboration between players can achieve our objective by taking, from each player distribution, a number of samples that is sufficient for learning the individual task, and then training a classifier over all samples. Such an algorithm uses k times as many samples as needed for learning an individual task — we say that this algorithm incurs $O(k)$ *overhead* in sample complexity. By contrast, we are interested in algorithms that take advantage of the collaborative environment, learn k tasks by sharing information, and incur

$o(k)$ overhead in sample complexity.

We study two variants of the aforementioned model: *personalized* and *centralized*. In the personalized setting (as in the department store example), we allow the learning algorithm to return different functions for different players. That is, our goal is to return classifiers $h_1, \ldots, h_k$ that have error of at most ϵ on player distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$, respectively. In the centralized setting (as in the hospital example), the learning algorithm is required to return a *single* classifier h that has an error of at most ϵ on all player distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$. Our results provide upper and lower bounds on the sample complexity overhead required for learning in both settings.

10.1.1 Overview of Results

In Section 10.3, we provide algorithms for personalized and centralized collaborative learning that obtain exponential improvements over the sample complexity of the naïve approach. In Theorem 10.3.1, we introduce an algorithm for the personalized setting that has $O(\ln(k))$ overhead in sample complexity. For the centralized setting, in Theorem 10.3.3, we develop an algorithm that has $O(\ln^2(k))$ overhead in sample complexity. At a high level, the latter algorithm first learns a series of functions on adaptively chosen mixtures of player distributions. These mixtures are chosen such that for any player a large majority of the functions perform well. This allows us to combine all functions into one classifier that performs well on every player distribution. Our algorithm is an improper learning algorithm, as the combination of these functions may not belong to $\mathcal{H}$.

In Section 10.4, we present lower bounds on the sample complexity of collaborative PAC learning for the personalized and centralized variants. In particular, in Theorem 10.4.1 we show that any algorithm that learns in the collaborative setting requires $\Omega(\ln(k))$ overhead in sample complexity. This shows that our upper bound for the personalized setting, as stated in Theorem 10.3.1, is tight. Furthermore, in Theorem 10.4.5, we show that obtaining *uniform convergence* across $\mathcal{H}$ over all k player distributions requires $\Omega(k)$ overhead in sample complexity. Interestingly, our centralized algorithm (Theorem 10.3.3) bypasses this lower bound by using arguments that do not depend on uniform convergence. Indeed, this can be seen from the fact that it is an improper learning algorithm.

In Section 10.5, we discuss the extension of our results to the non-realizable setting. Specifically, we consider a setting where there is a "good" but not "perfect" target function $h^* \in \mathcal{H}$ that has a small error with respect to every player distribution, and prove that our upper bounds carry over.

10.1.2 Related Work

Related work in computational and statistical learning has examined some aspects of the general problem of learning multiple related tasks simultaneously. Below we discuss papers on multi-task learning [43, 44, 45, 69, 186, 225], domain adaptation [46, 201, 202], and distributed learning [36, 99, 268], which are most closely related.

Multi-task learning considers the problem of learning multiple tasks in series or in parallel. In this space, Baxter [44] studied the problem of model selection for learning multiple related tasks. In their work, each learning task is itself randomly drawn from a distribution over related

tasks, and the learner's goal is to find a hypothesis space that is appropriate for learning all tasks. Ben-David and Schuller [45] also studied the sample complexity of learning multiple related tasks. However, in their work similarity between two tasks is represented by existence of "transfer" functions though which underlying distributions are related.

Mansour et al. [201, 202] consider a multi-source domain adaptation problem, where the learner is given k distributions and k corresponding predictors that have error at most ϵ on individual distributions. The goal of the learner is to combine these predictors to obtain error of $k\epsilon$ on any unknown mixture of player distributions. Our work is incomparable to this line of work, as our goal is to learn classifiers, rather than combining existing ones, and our benchmark is to obtain error ϵ on each individual distribution. Indeed, in our setting one can learn a hypothesis that has error $k\epsilon$ on any mixture of players with no overhead in sample complexity.

Distributed learning [36, 99, 268] also considers the problem of learning from k different distributions simultaneously. However, the main objective in this space is to learn with limited communication between the players, rather than with low sample complexity.

10.2 Model

Let $\mathcal{X}$ be an instance space and $\mathcal{Y} = \{-1, +1\}$ be the set of labels. A *hypothesis* is a function $h : \mathcal{X} \to \mathcal{Y}$ that maps any instance $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$. We consider a *hypothesis class* $\mathcal{H}$ with VC dimension d. Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the *error* of a hypothesis h is defined as $\mathrm{err}_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$.

In the *collaborative learning* setting, we consider k players with distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$ over $\mathcal{X} \times \mathcal{Y}$. We focus on the *realizable* setting, where all players' distributions are labeled according to a common target function $h^* \in \mathcal{H}$, i.e., $\mathrm{err}_{\mathcal{D}_i}(h^*) = 0$ for all $i \in [k]$ (but see Section 10.5 for an extension to the non-realizable setting). We represent an instance of the collaborative PAC learning setting with the 3-tuple $(\mathcal{H}, h^*, \{\mathcal{D}\}_{i\in[k]})$.

Our goal is to learn a good classifier with respect to *every* player distribution. We call this (ϵ, δ)-*learning* in the collaborative PAC setting, and study two variants: the *personalized* setting, and the *centralized* setting. In the personalized setting, our goal is to learn functions $h_1, \ldots, h_k$, such that with probability $1 - \delta$, $\mathrm{err}_{\mathcal{D}_i}(h_i) \leq \epsilon$ for all $i \in [k]$. In the centralized setting, we require all the output functions to be identical. Put another way, our goal is to return a single h, such that with probability $1 - \delta$, $\mathrm{err}_{\mathcal{D}_i}(h) \leq \epsilon$ for all $i \in [k]$. In both settings, we allow our algorithm to be *improper*, that is, the learned functions need not belong to $\mathcal{H}$.

We compare the sample complexity of our algorithms to their PAC counterparts in the realizable setting. In the traditional realizable PAC setting, $m_{\epsilon,\delta}$ denotes the number of samples needed for (ϵ, δ)-learning $\mathcal{H}$. That is, $m_{\epsilon,\delta}$ is the total number of samples drawn from a realizable distribution $\mathcal{D}$, such that, with probability $1 - \delta$, any classifier $h \in \mathcal{H}$ that is consistent with the sample set satisfies $\mathrm{err}_{\mathcal{D}}(h) \leq \epsilon$. We denote by $\mathcal{O}_{\mathcal{H}}(\cdot)$ the function that, for any set S of labeled samples, returns a function $h \in \mathcal{H}$ that is consistent with S if such a function exists (and outputs "none" otherwise). It is well-known that sampling a set S of size $m_{\epsilon,\delta} = O\left(\frac{1}{\epsilon}\left(d \ln\left(\frac{1}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right)\right)$, and applying $\mathcal{O}_{\mathcal{H}}(S)$, is sufficient for (ϵ, δ)-learning a hypothesis class $\mathcal{H}$ of VC dimension d [11]. We refer to the ratio of the sample complexity of an algorithm in the collaborative PAC setting to that of the (non-collaborative) realizable PAC setting as

the *overhead*. For ease of exposition, we only consider the dependence of the overhead on parameters k, d, and ϵ.

10.3 Sample Complexity Upper Bounds

In this section, we prove upper bounds on the sample complexity of (ϵ, δ)-learning in the collaborative PAC setting. We begin by providing a simple algorithm with $O(\ln(k))$ overhead (in terms of sample complexity, see Section 10.2) for the personalized setting. We then design and analyze an algorithm for the centralized setting with $O(\ln^2(k))$ overhead, following a discussion of additional challenges that arise in this setting.

10.3.1 Personalized Setting

The idea underlying the algorithm for the personalized setting is quite intuitive: If we were to learn a classifier that is on average good for the players, then we have learned a classifier that is good for a large fraction of the players. Therefore, a large fraction of the players can be simultaneously satisfied by a single good global classifier. This process can be repeated until each player receives a good classifier.

In more detail, let us consider an algorithm that pools together a sample set of total size $m_{\epsilon/4,\delta}$ from the uniform mixture $\mathcal{D} = \frac{1}{k}\sum_{i\in[k]} \mathcal{D}_i$ over individual player distributions, and finds $h \in \mathcal{H}$ that is consistent with this set. Clearly, with probability $1-\delta$, h has a small error of $\epsilon/4$ with respect to distribution $\mathcal{D}$. However, we would like to understand how well h performs on each individual player's distribution.

Since $\mathrm{err}_{\mathcal{D}}(h) \leq \epsilon/4$ is also the *average error* of h on player distributions, with probability $1-\delta$, h must have error of at most $\epsilon/2$ on at least half of the players. Indeed, one can identify such players by taking additional $\tilde{O}(\frac{1}{\epsilon})$ samples from each player and asking whether the empirical error of h on these sample sets is at most $3\epsilon/4$. Using a variant of the VC theorem, it is not hard to see that for any player i such that $\mathrm{err}_{\mathcal{D}_i}(h) \leq \epsilon/2$, the empirical error of h is at most $3\epsilon/4$, and no player with empirical error at most $3\epsilon/4$ has true error that is worst than ϵ. Once players with empirical error $3\epsilon/4$ are identified, one can output $h_i = f$ for any such player, and repeat the procedure for the remaining players. After $\log(k)$ rounds, this process terminates with all players having received functions with error of at most ϵ on their respective distributions, with probability $1 - \log(k)\delta$.

We formalize the above discussion via Algorithm 10.1 and Theorem 10.3.1. For completeness, a more rigorous proof of the theorem is given in Appendix A.

Theorem 10.3.1. *For any $\epsilon, \delta > 0$, and hypothesis class $\mathcal{H}$ of VC dimension d, Algorithm 10.1 (ϵ, δ)-learns $\mathcal{H}$ in the personalized collaborative PAC setting using m samples, where*

$$m = O\left(\frac{\ln(k)}{\epsilon}\left((d+k)\ln\left(\frac{1}{\epsilon}\right) + k\ln\left(\frac{k}{\delta}\right)\right)\right).$$

Note that when $k = O(d)$, this shows an overhead of $O(\ln(k))$.

Algorithm 10.1: PERSONALIZED LEARNING

Initialize $N_1 \leftarrow [k]$; $\delta' \leftarrow \delta/2\log(k)$.
for $r = 1, \ldots, \lceil \log(k) \rceil$ **do**
  $\tilde{\mathcal{D}}_r \leftarrow \frac{1}{|N_r|} \sum_{i \in N_r} \mathcal{D}_i$.
  Let S be a sample of size $m_{\epsilon/4,\delta'}$ drawn from $\tilde{\mathcal{D}}_r$, and $h^{(r)} \leftarrow \mathcal{O}_{\mathcal{H}}(S)$.
  Let $G_r \leftarrow \text{TEST}(h^{(r)}, N_r, \epsilon, \delta')$.
  $N_{r+1} \leftarrow N_r \setminus G_r$.
  for $i \in G_r$ **do** $h_i \leftarrow h^{(r)}$ **end for**
end for
return $h_1, \ldots, h_k$

Algorithm 10.2: $\text{TEST}(h, N, \epsilon, \delta)$

for $i \in N$ **do**
  Take sample set T_i of size $O\left(\frac{1}{\epsilon} \ln\left(\frac{|N|}{\epsilon\delta}\right)\right)$ from $\mathcal{D}_i$.
end for
return $\{i \mid \text{err}_{T_i}(h) \leq \frac{3}{4}\epsilon\}$.

Let us first introduce a lemma that shows that the procedure TEST in Algorithm 10.2 identifies the players who have a small error with respect to a given classifier. The proof of this lemma follows from the VC theorem and Chernoff bound (See Section 5.5 in [11]).

Lemma 10.3.2. *For any h, N, ϵ, and δ, with probability $1 - \delta$, $G = \text{TEST}(h, N, \epsilon, \delta)$ is such that*

1. *for any $i \in G$, $\text{err}_{\mathcal{D}_i}(h) \leq \epsilon$, and*
2. *for all $i \in N$, if $\text{err}_{\mathcal{D}_i}(h) \leq \frac{\epsilon}{2}$, then $i \in G$.*

We now use the above lemma to prove Theorem 10.3.1.

Proof of Theorem 10.3.1. Using Lemma 10.3.2, we have that for any iteration r of Algorithm 10.3, $G_r \leftarrow \text{TEST}(h^{(r)}, N_r, \epsilon, \delta')$ includes all players $i \in N_r$ for whom $\text{err}_{\mathcal{D}_i}(h^{(r)}) \leq \epsilon/2$ and no players $i \in N_r$ for whom $\text{err}_{\mathcal{D}_i}(h^{(r)}) > \epsilon$. Therefore, it is sufficient to prove that for every $r = 1, \ldots, \lceil \log(k) \rceil$, $|G_r| \geq \frac{1}{2}|N_r|$, in which case the algorithm ends after $\lceil \log(k) \rceil$ iterations and every player has received a function with error at most ϵ on his distribution.

In the remainder of the proof, we show that $|G_r| \geq \frac{1}{2}|N_r|$ for any r with probability $1 - \delta/\log(k)$. Recall that $h^{(r)}$ is learned by taking a sample of size $m_{\epsilon/4,\delta'}$ over distribution $\tilde{\mathcal{D}}_r = \frac{1}{|N_r|} \sum_{i \in N_r} \mathcal{D}_i$. Therefore, with probability $1 - \delta'$, $\text{err}_{\tilde{\mathcal{D}}_r}(h^{(r)}) \leq \epsilon/4$. By Markov's inequality, with probability $1 - \delta'$, $h^{(r)}$ has an error of $\epsilon/2$ for at least half of the players in N_r. Using Lemma 10.3.2, with probability at most δ', one or more such players are not included in G_r. Therefore, with overall probability $1 - 2\delta' = 1 - \delta/\log(k)$, $|G_r| \geq |N_r|/2$.

As for the sample complexity, TEST is called $\log(k)$ times, and requests

$$O\left(\frac{k\log(k)}{\epsilon} \cdot \ln\left(\frac{k}{\epsilon\delta}\right)\right) = O\left(\frac{\ln(k)}{\epsilon}\left(k\ln\left(\frac{1}{\epsilon}\right) + k\ln\left(\frac{k}{\delta}\right)\right)\right)$$

samples overall. Moreover, we learn a total of $\log(k)$ classifiers $h^{(r)}$, requesting

$$\log(k) \cdot m_{\epsilon/4,\delta'} = O\left(\frac{\log(k)}{\epsilon}\left(d\ln\left(\frac{1}{\epsilon}\right) + \ln\left(\frac{\log(k)}{\delta}\right)\right)\right).$$

samples overall. The sample complexity follows from these two bounds. □

10.3.2 Centralized Setting

We next present a learning algorithm with $O(\ln^2(k))$ overhead in the centralized setting. Recall that our goal is to learn a *single* function h that has an error of ϵ on every player distribution, as opposed to the personalized setting where players can receive different functions.

A natural first attempt at learning in the centralized setting is to combine the classifiers $h_1, \ldots, h_k$ that we learned in the personalized setting (Algorithm 10.1), say, through a weighted majority vote. One challenge with this approach is that, in general, it is possible that many of the functions h_j perform poorly on the distribution of a different player i. The reason is that when Algorithm 10.1 finds a suitable $h^{(r)}$ for players in G_r, it completely removes them from consideration for future rounds; subsequent functions may perform poorly with respect to the distributions associated with those players. Therefore, this approach may lead to a global classifier with large error on some player distributions.

To overcome this problem, we instead design an algorithm that continues to take additional samples from players for whom we have already found suitable classifiers. The key idea behind the centralized learning algorithm is to group the players at every round based on how many functions learned so far have large error rates on those players' distributions, and to learn from data sampled from all the groups simultaneously. This ensures that the function learned in each round performs well on a large fraction of the players in *each* group, thereby reducing the likelihood that in later stages of this process a player appears in a group for which a large fraction of the functions perform poorly.

In more detail, our algorithm learns $t = \Theta(\ln(k))$ classifiers $h^{(1)}, h^{(2)}, \ldots, h^{(t)}$, such that for any player $i \in [k]$, at least $0.6t$ functions among them achieve an error below $\epsilon' = \epsilon/6$ on $\mathcal{D}_i$. The algorithm then returns the classifier $\mathrm{maj}(\{h^{(r)}\}_{r=1}^{t})$, where, for a set of hypotheses H, $\mathrm{maj}(H)$ denotes the classifier that, given $x \in \mathcal{X}$, returns the label that the majority of hypotheses in H assign to x. Note that any instance that is mislabeled by this classifier must be mislabeled by at least $0.1t$ functions among the $0.6t$ good functions, i.e., $1/6$ of the good functions. Hence, $\mathrm{maj}(\{h^{(r)}\}_{r=1}^{t})$ has an error of at most $6\epsilon' = \epsilon$ on each distribution $\mathcal{D}_i$.

Throughout the algorithm, we keep track of counters $\alpha_i^{(r)}$ for any round $r \in [t]$ and player $i \in [k]$, which, roughly speaking, record the number of classifiers among $h^{(1)}, h^{(2)}, \ldots, h^{(r)}$ that have an error of more than ϵ' on distribution $\mathcal{D}_i$. To learn $h^{(r+1)}$, we first group distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$ based on the values of $\alpha_i^{(r)}$, draw about $m_{\epsilon',\delta}$ samples from the mixture of the distributions in each group, and return a function $h^{(r+1)}$ that is consistent with all of the samples. Similarly to Section 10.3.1, one can show that $h^{(r+1)}$ achieves $O(\epsilon')$ error with respect to a large fraction of player distributions in each group. Consequently, the counters are increased, i.e., $\alpha_i^{(r+1)} > \alpha_i^{(r)}$, only for a small fraction of players. Finally, we show that with high probability, $\alpha_i^{(t)} \leq 0.4t$ for any player $i \in [k]$, i.e., on each distribution $\mathcal{D}_i$, at least $0.6t$ functions achieve error of at most ϵ'.

The algorithm is formally described in Algorithm 10.3. The next theorem states our sample complexity upper bound for the centralized setting.

Algorithm 10.3: CENTRALIZED LEARNING

Set $\alpha_i^{(0)} \leftarrow 0$ for all $i \in [k]$, $t \leftarrow \left\lceil \frac{5}{2} \log_{8/7}(k) \right\rceil$, $\epsilon' \leftarrow \epsilon/6$, $N_0^{(0)} \leftarrow [k]$, $N_c^{(0)} \leftarrow \emptyset$ for each $c \in [t]$
for $r = 1, 2, \ldots, t$ **do**
  for $c = 0, 1, \ldots, t-1$ **do**
    if $N_c^{(r-1)} \neq \emptyset$ **then**
      Draw a sample set $S_c^{(r)}$ of size $m_{\epsilon'/16,\delta/(2t^2)}$ from $\widetilde{\mathcal{D}}_c^{(r-1)} = \frac{1}{|N_c^{(r-1)}|} \sum_{i \in N_c^{(r-1)}} \mathcal{D}_i$
    else $S_c^{(r)} \leftarrow \emptyset$.
    end if
  end for
  $h^{(r)} \leftarrow \mathcal{O}_{\mathcal{H}}\left(\bigcup_{c=0}^{t-1} S_c^{(r)}\right)$;
  $G_r \leftarrow \textsc{Test}(h^{(r)}, [k], \epsilon', \delta/(2t))$;
  for $i = 1, \ldots, k$ **do** $\alpha_i^{(r)} \leftarrow \alpha_i^{(r-1)} + \mathbb{I}\left(i \notin G_r\right)$ **end for**
  for $c = 0, \ldots, t$ **do** $N_c^{(r)} \leftarrow \{i \in [k] : \alpha_i^{(r)} = c\}$ **end for**
end for

Theorem 10.3.3. *For any $\epsilon, \delta > 0$, and hypothesis class $\mathcal{H}$ of VC dimension d, Algorithm 10.3 (ϵ, δ)-learns $\mathcal{H}$ in the centralized collaborative PAC setting using m samples, where*

$$m = O\left(\frac{\ln^2(k)}{\epsilon}\left((d+k)\ln\left(\frac{1}{\epsilon}\right) + k\ln\left(\frac{1}{\delta}\right)\right)\right).$$

In particular, Algorithm 10.3 has $O(\ln^2(k))$ overhead when $k = O(d)$.

Turning to the theorem's proof, note that in Algorithm 10.3, $N_c^{(r-1)}$ represents the set of players for whom c out of the $r-1$ functions learned so far have a large error, and $\widetilde{\mathcal{D}}_c^{(r-1)}$ represents the mixture of distribution of players in $N_c^{(r-1)}$. Moreover, G_r is the set of players for whom $h^{(r)}$ has a small error. The following lemma, whose proof appears in Appendix B.1, shows that with high probability each function $h^{(r)}$ has a small error on $\widetilde{\mathcal{D}}_c^{(r-1)}$ for all c. Here and in the following, t stands for $\left\lceil \frac{5}{2} \log_{8/7}(k) \right\rceil$ as in Algorithm 10.3.

Lemma 10.3.4. *With probability $1-\delta$, the following two properties hold for all $r \in [t]$:*

1. *For any $c \in \{0, \ldots, t-1\}$ such that $N_c^{(r-1)}$ is non-empty, $\mathrm{err}_{\widetilde{\mathcal{D}}_c^{(r-1)}}(h^{(r)}) \leq \epsilon'/16$.*
2. *For any $i \in G_r$, $\mathrm{err}_{\mathcal{D}_i}(h^{(r)}) \leq \epsilon'$, and for any $i \notin G_r$, $\mathrm{err}_{\mathcal{D}_i}(h^{(r)}) > \epsilon'/2$.*

Proof. Recall that for any r, c such that $N_c^{(r-1)}$ is non-empty, $h^{(r)}$ is consistent with the $m_{\epsilon'/16,\delta/(2t^2)}$ samples drawn from $\widetilde{\mathcal{D}}_c^{(r-1)}$. By the VC theorem, $\mathrm{err}_{\widetilde{\mathcal{D}}_c^{(r-1)}}(h^{(r)}) \leq \epsilon'/16$ holds with probability at least $1 - \delta/(2t^2)$. Also, by Lemma 10.3.2, the second statement holds with

probability $1 - \delta/(2t)$ for each $r \in [t]$. It follows from the union bound that with probability at least

$$1 - t^2 \cdot \delta/(2t^2) - t \cdot \delta/(2t) = 1 - \delta,$$

the two statements holds for any $r \in [t]$ simultaneously. □

The next lemma gives an upper bound on $|N_c^{(r)}|$ — the number of players for whom c out of the r learned functions have a large error.

Lemma 10.3.5. *With probability $1 - \delta$, for any $r, c \in \{0, \ldots, t\}$, we have $|N_c^{(r)}| \le \binom{r}{c} \cdot \frac{k}{8^c}$.*

Proof. Let $n_{r,c} = |N_c^{(r)}| = |\{i \in [k] : \alpha_i^{(r)} = c\}|$ be the number of players for whom c functions in $h^{(1)}, \ldots, h^{(r)}$ do not have a small error. We note that $n_{0,0} = k$ and $n_{0,c} = 0$ for $c \in \{1, \ldots, t\}$. The next technical claim asserts that to prove this lemma, it is sufficient to show that for any $r \in \{1, \ldots, t\}$ and $c \in \{0, \ldots, t\}$, $n_{r,c} \le n_{r-1,c} + \frac{1}{8} n_{r-1,c-1}$. Here we assume that $n_{r-1,-1} = 0$.

Lemma 10.3.6. *Suppose that $n_{0,0} = k$, $n_{0,c} = 0$ for $c \in \{1, \ldots, t\}$, and $n_{r,c} \le n_{r-1,c} + \frac{1}{8} n_{r-1,c-1}$ holds for any $r \in \{1, \ldots, t\}$ and $c \in \{0, \ldots, t\}$. Then for any $r, c \in \{0, \ldots, t\}$, $n_{r,c} \le \binom{r}{c} \cdot \frac{k}{8^c}$.*

Proof. We prove the claim by induction on r. Since $n_{0,0} = k$ and $n_{0,c} = 0$ for any $c \in [t]$, the inequality holds for $r = 0$. Suppose that for some $r \in [t]$, the inequality

$$n_{r',c} \le \binom{r'}{c} \cdot \frac{k}{8^c}$$

holds for $r' = r - 1$ and any $c \in \{0, 1, \ldots, t\}$. Then we have for any $c \in \{0, 1, \ldots, t\}$,

$$n_{r,c} \le n_{r-1,c} + n_{r-1,c-1}/8 \le \binom{r-1}{c} \cdot \frac{k}{8^c} + \binom{r-1}{c-1} \cdot \frac{k}{8^{c-1} \times 8} = \binom{r}{c} \cdot \frac{k}{8^c}.$$

Therefore, we conclude that the inequality holds for any $r, c \in \{0, 1, \ldots, t\}$. □

By definition of $\alpha_c^{(r)}$, $N_c^{(r)}$, and $n_{r,c}$, we have

$$\begin{aligned} n_{r,c} = \left|\{i \in [k] : \alpha_i^{(r)} = c\}\right| &\le \left|\{i \in [k] : \alpha_i^{(r-1)} = c\}\right| + \left|\{i \in [k] : \alpha_i^{(r-1)} = c - 1 \wedge i \notin G_r\}\right| \\ &= n_{r-1,c} + \left|N_{c-1}^{(r-1)} \setminus G_r\right|. \end{aligned}$$

It remains to show that $|N_{c-1}^{(r-1)} \setminus G_r| \le \frac{1}{8} n_{r-1,c-1}$. Recall that $\widetilde{\mathcal{D}}_{c-1}^{(r-1)}$ is the mixture of all distributions in $N_{c-1}^{(r-1)}$. By Lemma 10.3.4, with probability $1 - \delta$, $\operatorname{err}_{\widetilde{\mathcal{D}}_{c-1}^{(r-1)}}(h^{(r)}) < \epsilon'/16$. Put another way, $\sum_{i \in N_{c-1}^{(r-1)}} \operatorname{err}_{\mathcal{D}_i}(h^{(r)}) < \frac{\epsilon'}{16} \cdot |N_{c-1}^{(r-1)}|$. Thus, at most $\frac{1}{8}|N_{c-1}^{(r-1)}|$ players $i \in N_{c-1}^{(r-1)}$ can have $\operatorname{err}_{\mathcal{D}_i}(h^{(r)}) > \epsilon'/2$. Moreover, by Lemma 10.3.4, for any $i \notin G_r$, we have that $\operatorname{err}_{\mathcal{D}_i}(h^{(r)}) > \epsilon'/2$. Therefore,

$$\left|N_{c-1}^{(r-1)} \setminus G_r\right| \le \left|\{i \in N_{c-1}^{(r-1)} : \operatorname*{err}_{\mathcal{D}_i}(h^{(r)}) > \epsilon'/2\}\right| \le \frac{1}{8}\left|N_{c-1}^{(r-1)}\right| = \frac{1}{8} n_{r-1,c-1}.$$

This completes the proof. □

We now prove Theorem 10.3.3 using Lemma 10.3.5.

Proof of Theorem 10.3.3. We first show that, with high probability, for any $i \in [k]$, at most $0.4t$ functions among $h^{(1)}, \ldots, h^{(t)}$ have error greater than ϵ', i.e., $\alpha_i^{(t)} < 0.4t$ for all $i \in [k]$. Note that by our choice of $t = \left\lceil \frac{5}{2} \log_{8/7}(k) \right\rceil$, we have $(8/7)^{0.4t} \geq k$. By Lemma 10.3.5 and an upper bound on binomial coefficients, with probability $1 - \delta$, for any integer $c \in [0.4t, t]$,

$$|N_c^{(t)}| \leq \binom{t}{c} \cdot \frac{k}{8^c} < \left(\frac{et}{c}\right)^c \cdot \frac{k}{8^c} < \frac{k}{(8/7)^c} \leq 1,$$

which implies that $N_c^{(t)} = \emptyset$. Therefore, with probability $1 - \delta$, $\alpha_i^{(t)} < 0.4t$ for all $i \in [k]$.

Next, we prove that $f = \mathrm{maj}(\{h^{(r)}\}_{r=1}^t)$ has error at most ϵ on every player distribution. Consider distribution $\mathcal{D}_i$ of player i. By definition, $t - \alpha_i^{(t)}$ functions have error at most ϵ' on $\mathcal{D}_i$. We refer to these functions as "good" functions. Note that for any instance x that is mislabeled by h, at least $0.5t - \alpha_i^{(t)}$ good functions must make a wrong prediction. Therefore, $(t - \alpha_i^{(t)})\epsilon' \geq (0.5t - \alpha_i^{(t)}) \cdot \mathrm{err}_{\mathcal{D}_i}(h)$. Moreover, with probability $1 - \delta$, $\alpha_i^{(t)} < 0.4t$ for all $i \in [k]$. Hence,

$$\underset{\mathcal{D}_i}{\mathrm{err}}(h) \leq \frac{t - \alpha_i^{(t)}}{0.5t - \alpha_i^{(t)}} \epsilon' \leq \frac{0.6t}{0.1t} \epsilon' \leq \epsilon,$$

with probability $1 - \delta$. This proves that Algorithm 10.3 (ϵ, δ)-learns $\mathcal{H}$ in the centralized collaborative PAC setting.

Finally, we bound the sample complexity of Algorithm 10.3. Recall that $t = \Theta(\ln(k))$ and $\epsilon' = \epsilon/6$. At each iteration of Algorithm 10.3, we draw total of $t \cdot m_{\epsilon'/16, \delta/(4t^2)}$ samples from t mixtures. Therefore, over t time steps, we draw a total of

$$t^2 \cdot m_{\epsilon'/16, \delta/(4t^2)} = O\left(\frac{\ln^2(k)}{\epsilon} \cdot \left(d \ln\left(\frac{1}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right) + \ln\ln(k)\right)\right)$$

samples for learning $h^{(1)}, \ldots, h^{(t)}$. Moreover, the total number samples requested for subroutine $\textsc{Test}(h^{(r)}, [k], \epsilon', \delta/(4t))$ for $r = 1 \ldots, t$ is

$$O\left(\frac{tk}{\epsilon} \cdot \ln\left(\frac{k}{\epsilon\delta}\right)\right) = O\left(\frac{\ln(k)}{\epsilon} \cdot \left(k \ln\left(\frac{1}{\epsilon}\right) + k \ln\left(\frac{1}{\delta}\right)\right) + \frac{\ln^2(k)}{\epsilon} k\right).$$

We conclude that the total sample complexity is

$$O\left(\frac{\ln^2(k)}{\epsilon}\left((d+k)\ln\left(\frac{1}{\epsilon}\right) + k \ln\left(\frac{1}{\delta}\right)\right)\right).$$

□

10.4 Sample Complexity Lower Bounds

In this section, we present lower bounds on the sample complexity of collaborative PAC learning. In Section 10.4.1, we show that any learning algorithm for the collaborative PAC setting incurs $\Omega(\log(k))$ overhead in terms of sample complexity. In Section 10.4.2, we consider the sample complexity required for obtaining *uniform convergence* across $\mathcal{H}$ in the collaborative PAC setting. We show that $\Omega(k)$ overhead is necessary to obtain such results.

10.4.1 Tight Lower Bound for the Personalized Setting

We now turn to establishing the $\Omega(\log(k))$ lower bound mentioned above. This lower bound implies the tightness of the $O(\log(k))$ overhead upper bound obtained by Theorem 10.3.1 in the personalized setting. Moreover, the $O(\log^2(k))$ overhead obtained by Theorem 10.3.3 in the centralized setting is nearly tight, up to a $\log(k)$ multiplicative factor. Formally, we prove the following theorem.

Theorem 10.4.1. *For any $k \in \mathbb{N}$, $\epsilon, \delta \in (0, 0.1)$, and (ϵ, δ)-learning algorithm $\mathcal{A}$ in the collaborative PAC setting, there exist an instance with k players, and a hypothesis class of VC-dimension k, on which $\mathcal{A}$ requires at least $3k \ln[9k/(10\delta)]/(20\epsilon)$ samples in expectation.*

Hard instance distribution

We show that for any $k \in \mathbb{N}$ and $\epsilon, \delta \in (0, 0.1)$, there is a distribution $\boldsymbol{\mathcal{D}}_{k,\epsilon}$ of "hard" instances, each with k players and a hypothesis class with VC-dimension k, such that any (ϵ, δ)-learning algorithm $\mathcal{A}$ requires $\Omega(k \log(k)/\epsilon)$ samples in expectation on a random instance drawn from the distribution, even in the personalized setting. This directly implies Theorem 10.4.1, since $\mathcal{A}$ must take $\Omega(k \log(k)/\epsilon)$ samples on some instance in the support of $\boldsymbol{\mathcal{D}}_{k,\epsilon}$. We define $\boldsymbol{\mathcal{D}}_{k,\epsilon}$ as follows:

- Instance space: $\mathcal{X}_k = \{1, 2, \ldots, k, \perp\}$.
- Hypothesis class: $\mathcal{H}_k$ is the collection of all binary functions on $\mathcal{X}_k$ that map $\perp$ to 0.
- Target function: h^* is chosen from $\mathcal{H}_k$ uniformly at random.
- Players' distributions: The distribution $\mathcal{D}_i$ of player i is either a degenerate distribution that assigns probability 1 to $\perp$, or a Bernoulli distribution on $\{i, \perp\}$ with $\mathcal{D}_i(i) = 2\epsilon$ and $\mathcal{D}_i(\perp) = 1 - 2\epsilon$. $\mathcal{D}_i$ is chosen from these two distributions independently and uniformly at random.

Note that the VC-dimension of $\mathcal{H}_k$ is k. Moreover, on any instance in the support of $\boldsymbol{\mathcal{D}}_{k,\epsilon}$, learning in the personalized setting is equivalent to learning in the centralized setting. This is due to the fact that given functions $h_1, h_2, \ldots, h_k$ for the personalized setting, where h_i is the function assigned to player i, we can combine these functions into a single function $h \in \mathcal{H}_k$ for the centralized setting by defining $h(\perp) = 0$ and $h(i) = h_i(i)$ for all $i \in [k]$. Then, $\mathrm{err}_{\mathcal{D}_i}(h) \leq \mathrm{err}_{\mathcal{D}_i}(h_i)$ for all $i \in [k]$.[1] Therefore, without loss of generality we focus below on the centralized setting.

Lower bound for $k = 1$

As a building block in our proof of Theorem 10.4.1, we establish a lower bound for the special case of $k = 1$. For brevity, let $\boldsymbol{\mathcal{D}}_\epsilon$ denote the instance distribution $\boldsymbol{\mathcal{D}}_{1,\epsilon}$. We say that $\mathcal{A}$ is an (ϵ, δ)-learning algorithm for the instance distribution $\boldsymbol{\mathcal{D}}_\epsilon$ if and only if on any instance in the

[1] In fact, when $h_i \in \mathcal{H}_k$, $\mathrm{err}_{\mathcal{D}_i}(h) = \mathrm{err}_{\mathcal{D}_i}(h_i)$ for all $i \in [k]$.

support of $\boldsymbol{\mathcal{D}}_\epsilon$, with probability at least $1-\delta$, $\mathcal{A}$ outputs a function h with error below ϵ. The following lemma states that any (ϵ,δ)-learning algorithm for $\boldsymbol{\mathcal{D}}_\epsilon$ takes $\Omega(\log(1/\delta)/\epsilon)$ samples on a random instance drawn from $\boldsymbol{\mathcal{D}}_\epsilon$.[2]

Lemma 10.4.2. *For any $\epsilon,\delta\in(0,0.1)$ and (ϵ,δ)-learning algorithm $\mathcal{A}$ for $\boldsymbol{\mathcal{D}}_\epsilon$, $\mathcal{A}$ takes at least $\ln(1/\delta)/(6\epsilon)$ samples in expectation on a random instance drawn from $\boldsymbol{\mathcal{D}}_\epsilon$. Here the expectation is taken over both the randomness in the samples and the randomness in drawing the instance from $\boldsymbol{\mathcal{D}}_\epsilon$.*

Proof. Recall that in any instance in the support of $\boldsymbol{\mathcal{D}}_\epsilon$, the instance space is $\mathcal{X}=\{1,\perp\}$, while the hypothesis class is $\mathcal{H}=\{h_0,h_1\}$, where $h_i(\perp)=0$ and $h_i(1)=i$ for $i\in\{0,1\}$.

Fix an (ϵ,δ)-learning algorithm $\mathcal{A}$ for distribution $\boldsymbol{\mathcal{D}}_\epsilon$. Suppose $\mathcal{A}$ runs on an instance drawn from $\boldsymbol{\mathcal{D}}_\epsilon$ with a degenerate distribution $\mathcal{D}$. For $i\in\{0,1\}$, let $\mathcal{E}_i$ denote the event that $\mathcal{A}$ outputs h_i. Define random variable T as the number of samples drawn by $\mathcal{A}$ before it terminates, and let $p_n=\Pr[T=n|\mathcal{E}_0]$.

Now we consider the situation that $\mathcal{A}$ runs on the instance with $\mathcal{D}'(1)=2\epsilon$, $\mathcal{D}'(\perp)=1-2\epsilon$, and the target function is h_1. On this instance, with probability at least $p_n\cdot(1-2\epsilon)^n$, $\mathcal{A}$ outputs h_0 after drawing exactly n samples, all of which are $\perp$. Since $\mathcal{D}'(1)=2\epsilon$ and the target function is h_1, $\mathrm{err}_{\mathcal{D}'}(h_0)=2\epsilon>\epsilon$, i.e., $\mathcal{A}$ outputs a function with error greater than ϵ on $\mathcal{D}'$. Since $\mathcal{A}$ is an (ϵ,δ)-learning algorithm for $\boldsymbol{\mathcal{D}}_\epsilon$,

$$\delta\geq\sum_{n=0}^{\infty}p_n\cdot(1-2\epsilon)^n=\mathbb{E}\left[(1-2\epsilon)^T\,\middle|\,\mathcal{E}_0\right].$$

By Jensen's inequality and the convexity of the function $\log_{1-2\epsilon}x$ for $\epsilon\in(0,0.1)$, we have

$$\mathbb{E}\left[T|\mathcal{E}_0\right]\geq\log_{1-2\epsilon}\mathbb{E}\left[(1-2\epsilon)^T|\mathcal{E}_0\right]\geq\log_{1-2\epsilon}\delta=\frac{\ln(1/\delta)}{\ln[1/(1-2\epsilon)]}\geq\frac{\ln(1/\delta)}{3\epsilon}.$$

Here the last step holds since $\ln[1/(1-2\epsilon)]\leq 3\epsilon$ for any $\epsilon\in(0,0.1)$. A similar argument (using the same distribution $\mathcal{D}'$, but the target function h_0 instead of h_1) gives $\mathbb{E}\left[T|\mathcal{E}_1\right]\geq\ln(1/\delta)/(3\epsilon)$.

Therefore, $\mathcal{A}$ takes at least $\ln(1/\delta)/(3\epsilon)$ samples in expectation when the distribution $\mathcal{D}$ is degenerate, which happens with probability $1/2$ for an instance drawn from $\boldsymbol{\mathcal{D}}_\epsilon$. Therefore, the expected sample complexity of $\mathcal{A}$ on a random instance sampled from $\boldsymbol{\mathcal{D}}_\epsilon$ is lower bounded by

$$\frac{1}{2}\cdot\frac{\ln(1/\delta)}{3\epsilon}=\frac{\ln(1/\delta)}{6\epsilon}.$$

□

Lower bound for general k

Now we prove Theorem 10.4.1 by Lemma 10.4.2 and a reduction from a random instance sampled from $\boldsymbol{\mathcal{D}}_\epsilon$ to instances sampled from $\boldsymbol{\mathcal{D}}_{k,\epsilon}$. Intuitively, a random instance drawn from

[2] Here we only assume that $\mathcal{A}$ is correct for instances in the support of $\boldsymbol{\mathcal{D}}_\epsilon$, rather than being correct on *every instance*.

$\boldsymbol{\mathcal{D}}_{k,\epsilon}$ is equivalent to k independent instances from $\boldsymbol{\mathcal{D}}_\epsilon$. We show that any learning algorithm $\mathcal{A}$ that simultaneously solves k tasks (i.e., an instance from $\boldsymbol{\mathcal{D}}_{k,\epsilon}$) with probability $1-\delta$ can be transformed into an algorithm $\mathcal{A}'$ that solves a single task (i.e., an instance from $\boldsymbol{\mathcal{D}}_\epsilon$) with probability $1-O(\delta/k)$. Moreover, the expected sample complexity of $\mathcal{A}'$ is only an $O(1/k)$ fraction of the complexity of $\mathcal{A}$. This transformation, together with Lemma 10.4.2, gives a lower bound on the sample complexity of $\mathcal{A}$.

Proof of Theorem 10.4.1. We construct an algorithm $\mathcal{A}'$ for the instance distribution $\boldsymbol{\mathcal{D}}_\epsilon$ from an algorithm $\mathcal{A}$ that (ϵ,δ)-learns in the centralized setting. Recall that on an instance drawn from $\boldsymbol{\mathcal{D}}_\epsilon$, $\mathcal{A}'$ has access to a distribution $\mathcal{D}$, i.e., the single player's distribution.

- $\mathcal{A}'$ generates an instance $(\mathcal{H}_k, h^*, \{\mathcal{D}_i\}_{i\in[k]})$ from the distribution $\boldsymbol{\mathcal{D}}_{k,\epsilon}$ (specifically, $\mathcal{A}'$ knows the target function h^* and the distributions), and then chooses $l \in [k]$ uniformly at random.
- $\mathcal{A}'$ simulates $\mathcal{A}$ on instance $(\mathcal{H}_k, h^*, \{\mathcal{D}_i\}_{i\in[k]})$, with $\mathcal{D}_l$ replaced by the distribution $\mathcal{D}$. Specifically, every time $\mathcal{A}$ draws a sample from $\mathcal{D}_j$ for some $j \neq l$, $\mathcal{A}'$ samples $\mathcal{D}_j$ and forwards the sample to $\mathcal{A}$. When $\mathcal{A}$ asks for a sample from $\mathcal{D}_l$, $\mathcal{A}'$ samples the distribution $\mathcal{D}$ instead and replies to $\mathcal{A}$ accordingly, i.e., $\mathcal{A}'$ returns l, together with the label, if the sample is 1 (recall that $\mathcal{X}_1 = \{1, \perp\}$), and returns $\perp$ otherwise.
- Finally, when $\mathcal{A}$ terminates and returns a function h on $\mathcal{X}_k$, $\mathcal{A}'$ checks whether $\mathrm{err}_{\mathcal{D}_j}(h) < \epsilon$ for every $j \neq l$. If so, $\mathcal{A}'$ returns the function f' defined as $h'(1) = h(l)$ and $h'(\perp) = h(\perp)$. Otherwise, $\mathcal{A}'$ repeats the simulation process on a new instance drawn from $\boldsymbol{\mathcal{D}}_{k,\epsilon}$.

Let m_i be the expected number of samples drawn from the i-th distribution when $\mathcal{A}$ runs on an instance drawn from $\boldsymbol{\mathcal{D}}_{k,\epsilon}$.

Claim 10.4.3. *$\mathcal{A}'$ is an $(\epsilon, 10\delta/(9k))$-learning algorithm for $\boldsymbol{\mathcal{D}}_\epsilon$.*

Proof of Claim 10.4.3. Let p_i be the probability that, on a random instance drawn from $\boldsymbol{\mathcal{D}}_{k,\epsilon}$, the function f returned by $\mathcal{A}$ satisfies $\mathrm{err}_{\mathcal{D}_i}(h) > \epsilon$ and $\mathrm{err}_{\mathcal{D}_j}(h) \leq \epsilon$ for any $j \neq i$. By assumption, $\sum_{i=1}^k p_i \leq \delta$.

Let random variable T denote the number of times that $\mathcal{A}'$ repeats the simulation process (it repeats the process every time the condition $\forall j \neq l,\ \mathrm{err}_{\mathcal{D}_j}(h) < \epsilon$ is violated). Let $\mathcal{E}_i$ denote the event that $\mathcal{A}'$ returns a function with error greater than ϵ and $T = i$. Clearly, $\mathcal{E}_i$ implies:

1. The simulated algorithm $\mathcal{A}$ fails to return a function with an error smaller than ϵ on every distribution in each of the first $i-1$ simulations, which happens with probability at most δ^{i-1}.
2. In the i-th iteration, $\mathcal{A}$ returns a function f such that $\mathrm{err}_{\mathcal{D}_j}(h) \leq \epsilon$ for $j \neq l$, yet $\mathrm{err}_{\mathcal{D}_l}(h) > \epsilon$. This happens with probability p_l.

Recall that l is drawn uniformly at random from $[k]$. Thus,

$$\Pr[\mathcal{E}_i] \leq \delta^{i-1} \cdot \frac{1}{k} \sum_{i=1}^{k} p_i \leq \delta^i/k.$$

Overall, the probability that $\mathcal{A}'$ returns a function with error greater than ϵ is bounded by

$$\sum_{i=1}^{\infty} \Pr[\mathcal{E}_i] \leq \sum_{i=1}^{\infty} \delta^i/k = \frac{\delta}{k(1-\delta)} \leq \frac{10\delta}{9k}.$$

which proves that $\mathcal{A}'$ is an $(\epsilon, 10\delta/(9k))$-learning algorithm for $\boldsymbol{\mathcal{D}}_\epsilon$. □

Claim 10.4.4. *$\mathcal{A}'$ takes at most $10/(9k)\sum_{i=1}^{k} m_i$ samples in expectation.*

Proof of Claim 10.4.4. Let random variable T denote the number of times that $\mathcal{A}'$ repeats the simulation process. Let $X_1, X_2, \ldots$ be the number of samples drawn from distribution $\mathcal{D}$ in each simulation. Note that these random variables are independently and identically distributed, so by Wald's equation,

$$\mathbb{E}\left[\sum_{i=1}^{T} X_i\right] = \mathbb{E}[T] \cdot \mathbb{E}[X_1].$$

For any positve integer i, $T \geq i$ holds only if the simulated algorithm $\mathcal{A}$ fails to return a function with an error smaller than ϵ on every distribution in each of the first $i-1$ simulations. By assumption, this happens with probability at most δ^{i-1}. Therefore,

$$\mathbb{E}[T] = \sum_{i=1}^{\infty} \Pr[T \geq i] \leq \sum_{i=0}^{\infty} \delta^i \leq \frac{1}{1-\delta} \leq \frac{10}{9}.$$

Note that conditioning on the value of l in the first iteration of $\mathcal{A}'$, $\mathcal{A}'$ draws m_l samples from $\mathcal{D}$ in expectation. Since l is uniformly distributed in $[k]$,

$$\mathbb{E}[X_1] = \frac{1}{k}\sum_{i=1}^{k} m_i,$$

and the expected number of samples taken by $\mathcal{A}'$ in total is at most

$$\mathbb{E}[T] \cdot \mathbb{E}[X_1] \leq \frac{10}{9k}\sum_{i=1}^{k} m_i.$$

□

Applying Lemma 10.4.2 to $\mathcal{A}'$ gives $\sum_{i=1}^{k} m_i \geq \frac{3k\ln[9k/(10\delta)]}{20\epsilon}$, which proves Theorem 10.4.1. □

10.4.2 Lower Bound for Uniform Convergence

We next examine the sample complexity required for obtaining *uniform convergence* across the hypothesis class $\mathcal{H}$ in the centralized collaborative PAC setting, and establish an overhead lower bound of $\Omega(k)$. Interestingly, our centralized learning algorithm (Algorithm 10.3) achieves $O(\log^2(k))$ overhead — it circumvents the lower bound by not relying on uniform convergence.

To be more formal, we first need to define uniform convergence in the cooperative PAC learning setting. We say that a hypothesis class $\mathcal{H}$ has the *uniform convergence property* with sample size $m_{\epsilon,\delta}^{(k)}$ if for any k distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$, there exist integers $m_1, \ldots, m_k$ that sum up to $m_{\epsilon,\delta}^{(k)}$, such that when m_i samples are drawn from $\mathcal{D}_i$ for each $i \in [k]$, with probability $1-\delta$, any function in $\mathcal{H}$ that is consistent with all the $m_{\epsilon,\delta}^{(k)}$ samples achieves error at most ϵ on every distribution $\mathcal{D}_i$.

Note that the foregoing definition is a relatively weak adaptation of uniform convergence to the cooperative setting, as the integers m_i are allowed to depend on the distributions $\mathcal{D}_i$. But this observation only strengthens our lower bound, which holds despite the weak requirement.

Theorem 10.4.5. *For any $k, d \in \mathbb{N}$ and $(\epsilon, \delta) \in (0, 0.1)$, there exists a hypothesis class $\mathcal{H}$ of VC-dimension d, such that $m_{\epsilon,\delta}^{(k)} \geq dk(1-\delta)/(4\epsilon)$.*

Proof. Fix $k, d \in \mathbb{N}$ and $\epsilon, \delta \in (0, 0.1)$. We define instance $(\mathcal{H}, h^*, \{\mathcal{D}_i\}_{i=1}^k)$ as follows:

- Instance space: $\mathcal{X} = ([k] \times [d]) \cup \{\perp\}$.
- Hypothesis class: $\mathcal{H}$ is the collection of all binary functions on $\mathcal{X}$ that map $\perp$ to 0 and take value 1 on at most d points.
- Target function: h^* maps every element in $\mathcal{X}$ to 0.
- Players' distributions: for each player $i \in [k]$, $\mathcal{D}_i((i,j)) = 2\epsilon/d$ for any $j \in [d]$ and $\mathcal{D}_i(\perp) = 1 - 2\epsilon$.

Let $m_1, m_2, \ldots, m_k$ be integers such that $m_1 + m_2 + \cdots + m_k = m_{\epsilon,\delta}^{(k)}$, and when m_i samples are drawn from $\mathcal{D}_i$ for each $i \in [k]$, with probability $1-\delta$, any consistent function in $\mathcal{H}$ has an error at most ϵ on every $\mathcal{D}_i$. We consider the following algorithm $\mathcal{A}$ that proceeds in rounds: in each round, $\mathcal{A}$ draws m_i samples from $\mathcal{D}_i$ for each $i \in [k]$. $\mathcal{A}$ terminates if at the end of some round, $\mathrm{err}_{\mathcal{D}_i}(h) \leq \epsilon$ for all $i \in [k]$ and any function $h \in \mathcal{H}$ that is consistent with the $m_{\epsilon,\delta}^{(k)}$ samples. In expectation, $\mathcal{A}$ terminates after at most $1/(1-\delta)$ rounds, and takes at most $m_{\epsilon,\delta}^{(k)}/(1-\delta)$ samples.

Note that if a sample set contains strictly less than $d/2$ elements in $\{(i^*, 1), (i^*, 2), \ldots, (i^*, d)\}$ for some i^*, there is a consistent function in $\mathcal{H}$ with error strictly above ϵ on $\mathcal{D}_{i^*}$, namely, the function that maps (i, j) to 1 if and only if $i = i^*$ and (i^*, j) is not in the sample set. Therefore, when $\mathcal{A}$ terminates, at least $d/2$ elements from $\mathcal{X} \setminus \{\perp\}$ have been drawn from each distribution.

Note that the probability that each sample is different from $\perp$ is 2ϵ, so, in expectation, $(d/2) \cdot (1/(2\epsilon)) = d/(4\epsilon)$ samples from each distribution are required to draw $d/2$ samples from $\mathcal{X} \setminus \{\perp\}$. Therefore, we have $m_{\epsilon,\delta}^{(k)}/(1-\delta) \geq dk/(4\epsilon)$, which proves the theorem. □

10.5 Extension to the Non-realizable Setting

In this section, we generalize our sample complexity upper bounds in Section 10.3 to the *non-realizable setting*, where we have a weaker assumption on the consistency between players'

distributions. Instead of assuming a perfect target function in $\mathcal{H}$ with zero error on every distribution, we consider the case that there exists $h^* \in \mathcal{H}$ with $\mathrm{err}_{\mathcal{D}_i}(h^*) \leq \epsilon/100$ for all $i \in [k]$. Our goal is still to output a single function or multiple functions, such that the function assigned to each player has an error below ϵ on that player's distribution. We call this the *non-realizable collaborative PAC setting*, and prove analogues of Theorems 10.3.1 and 10.3.3.

Theorem 10.5.1. *There is an (ϵ, δ)-learning algorithm in the non-realizable personalized collaborative PAC setting using m samples, where*

$$m = O\left(\frac{\log(k)}{\epsilon}\left((d+k)\log\left(\frac{1}{\epsilon}\right) + k\log\left(\frac{k}{\delta}\right)\right)\right).$$

Theorem 10.5.2. *There is an (ϵ, δ)-learning algorithm in the non-realizable centralized collaborative PAC setting using m samples, where*

$$m = O\left(\frac{\log^2(k)}{\epsilon}\left((d+k)\ln\left(\frac{1}{\epsilon}\right) + k\ln\left(\frac{1}{\delta}\right)\right)\right).$$

We prove Theorem 10.5.2 by slightly adapting Algorithm 10.3; Theorem 10.5.1 can be proved similarly.

Proof of Theorem 10.5.2. Recall that in each round r of Algorithm 10.3, we draw

$$m_{\epsilon'/16,\delta/(2t^2)} = m_{\epsilon/96,\delta/(2t^2)}$$

samples from each mixture $\widetilde{\mathcal{D}}_c^{(r-1)}$ and query the oracle $\mathcal{O}_{\mathcal{H}}$ on the union of all the samples. Now suppose that, instead, we draw from $\widetilde{\mathcal{D}}_c^{(r-1)}$ a dataset $S_c^{(r)}$ of size

$$C \cdot \frac{d\ln(1/\epsilon) + \ln(2t^2/\delta)}{\epsilon}.$$

By [11, Theorem 5.7], we can choose a sufficiently large constant C such that with probability $1 - \delta/(2t^2)$, for any function $h \in \mathcal{H}$:

1. $\mathrm{err}_{\widetilde{\mathcal{D}}_c^{(r-1)}}(h) \leq \epsilon/100$ implies $\mathrm{err}_{S_c^{(r)}}(h) \leq \epsilon/98$.
2. $\mathrm{err}_{\widetilde{\mathcal{D}}_c^{(r-1)}}(h) > \epsilon/96$ implies $\mathrm{err}_{S_c^{(r)}}(h) > \epsilon/98$.

Then we choose $h^{(r)}$ such that the empirical error of $h^{(r)}$ on every dataset $S_c^{(r)}$ is upper bounded by $\epsilon/98$. Note that given that the two conditions above hold, such a function $h^{(r)}$ always exists, since we assume that $\mathrm{err}_{\widetilde{\mathcal{D}}_c^{(r-1)}}(h^*) \leq \epsilon/100$. Other parts of Algorithm 10.3 remain unchanged.

The modified algorithm indeed (ϵ, δ)-learns in the non-realizable collaborative PAC setting. Since we guarantee that with probability $1 - \delta/(2t^2)$, the error of $h^{(r)}$ on $\widetilde{\mathcal{D}}_c^{(r)}$ is bounded by $\epsilon/96$, Lemma 10.3.4 and the rest of the proof still holds. Furthermore, the (asymptotic) sample complexity of the algorithm does not change, as the number of samples drawn from each mixture only increases by a constant factor. □

10.6 Discussion and Subsequent Works

We remark that Algorithm 10.3 is inspired by the classic boosting scheme. Indeed, an algorithm that is directly adapted from boosting attains a similar performance guarantee as in Theorem 10.3.3. The algorithm assigns a uniform weight to each player, and learns a classifier with $O(\epsilon)$ error on the mixture distribution. Then, depending on whether the function achieves an $O(\epsilon)$ error on each distribution, the algorithm updates the players' weights, and learns the next classifier from the weighted mixture of all distributions. An analysis similar to that of AdaBoost [123] shows that the majority vote of all the classifiers learned over $\Theta(\ln(k))$ iterations of the above procedure achieves a small error on *every* distribution. However, similar to Algorithm 10.3, this algorithm achieves an $O(\ln^2(k))$ overhead for the centralized setting.

Following the initial publication of our results, Chen et al. [76] considered the same weighting scheme inspired by boosting and Multiplicative Weight Update. They showed that by coupling this weighting scheme with changes to TEST (Algorithm 10.2)—which requires fewer samples, but may occasionally be wrong—they have a centeralized collaborative PAC learning algorithm that has $O(\ln(k))$ overhead.

Part IV

Learning for People

Chapter 11

A Near Optimal Kidney Exchange with a Few Queries

11.1 Introduction

The best treatment for people who suffer from chronic kidney disease is transplanting a healthy kidney. Transplanted kidneys are usually harvested from deceased donors; but as of June 2018, there are 114877 people on the U.S. national waiting list [261], making the median waiting time dangerously long. Fortunately, kidneys are an unusual organ in that donation by living donors is also a possibility—as long as patients happen to be medically compatible with their potential donors.

In its simplest form—*pairwise exchange*—two incompatible donor-patient pairs exchange kidneys: the donor of the first pair donates to the patient of the second pair, and the donor of the second pair donates to the patient of the first pair. This setting can be represented as an undirected *compatibility graph*, where each vertex represents an incompatible donor-patient pair, and an edge between two vertices represents the possibility of a pairwise exchange. A matching in this graph specifies which exchanges take place. Modern kidney exchange programs regularly employ swaps involving *three* donor-patient pairs, which are known to provide significant benefit compared to pairwise swaps alone [14, 234].

The edges of the compatibility graph can be determined based on the medical characteristics—blood type and tissue type—of donors and patients. However, the compatibility graph only tells part of the story. Before a transplant takes place, a more accurate medical test known as a *crossmatch test* takes place. This test involves mixing samples of the blood of the patient and the donor (rather than simply looking up information in a database), making the test relatively costly and time consuming. Moreover, crossmatch test may can fail with significant probability. Of interest, then are procedures that perform crossmatches for a subset of donor-patient pairs to find ones that are compatible, and based on these tests, match as many donor-patient pairs as possible.

Motivated by this, we consider the *stochastic matching* problem. In this problem, we are given an undirected graph $G = (V, E)$, where we do not know which edges in E actually exist. Rather, for each edge $e \in E$, we are given an existence probability p_e. Our goal is to find

algorithms that first query some subset of edges to find ones that exist, and based on these queries, produce a matching that is as large as possible. The stochastic matching problem is a special case of *stochastic k-cycle packing*, where each cycle exists only when all of its edges exists, and the goal is to find a (vertex disjoint) packing of existing cycles that collectively cover the maximum number of vertices possible.

Without any constraints, one can simply query all edges, and then output the maximum matching or packing over those that exist—hereafter, referred to as the omniscient optimal solution. But this level of freedom may not always be available, therefore, we are interested in the tradeoff between the number of queries and the fraction of the omniscient optimal solution achieved. Specifically, we ask: In order to perform as well as the omniscient optimum in the stochastic matching problem, do we need to query (almost) all the edges, that is, do we need a budget of $\Theta(n)$ queries per vertex, where n is the number of vertices? Or, can we, for any arbitrarily small $\epsilon > 0$, achieve a $(1-\epsilon)$ fraction of the omniscient optimum by using an $o(n)$ per-vertex budget? We answer these questions, as well as their extensions to the k-cycle packing problem. We support our theoretical results empirically on both generated and real data from a large fielded kidney exchange in the United States.

11.1.1 Our theoretical results and techniques

Our main theoretical result gives a positive answer to the latter question for stochastic matching, by showing that, surprisingly, a *constant* per-vertex budget is sufficient to get ϵ-close to the omniscient optimum. Indeed, we design a polynomial-time algorithm with the following properties: for any constant $\epsilon > 0$, the algorithm queries at most $O_\epsilon(1)$ edges incident to any particular vertex, requires $O_\epsilon(1)$ rounds of parallel queries, and achieves $(1-\epsilon)$ fraction of the omniscient optimum. This guarantee holds as long as all the non-zero p_e's are bounded away from zero by some constant that is independent of n (See Section 11.9 for a discussion of cases where p_e may be arbitrarily small for a few edges).[1]

The foregoing algorithm is *adaptive*, in the sense that its queries are conditioned on the answers to previous queries. Even though it requires only a constant number of rounds, it is natural to ask whether a non-adaptive algorithm—one that issues all its queries in one round—can also achieve a similar guarantee. We do not give a complete answer to this question, but we do present a non-adaptive algorithm that achieves a $0.5(1-\epsilon)$-approximation (for arbitrarily small $\epsilon > 0$) to the omniscient optimum.

We also extend our results to the stochastic k-cycle packing problem, where we are given a directed graph and the collection of all of its cycles of length at most k. The goal is to find the collection of mutually vertex-disjoint cycles, called a packing, that covers the maximum number of vertices possible. Stochastic matching is a special case of stochastic k-cycle packing: each undirected edge in stochastic matching corresponds to a cycle of length 2, that is, $k = 2$. In stochastic k-cycle packing, each cycle exists if and only if all of its edges exist. That is, when p represents the probability of a directed edge existing, then a cycle of length ℓ exists with probability p^ℓ, although these events are correlated across cycles that share an edge. Our goal is

[1]Notation $O_\epsilon(1)$ refers to asymptothic behavior that is constant when ϵ is a fixed constant. When it is clear from the context, we use $O(1)$ instead of $O_\epsilon(1)$.

to query the edges and output a collection of existing *vertex-disjoint* cycles that covers a large number of vertices. We present an adaptive polynomial-time algorithms that for any constant $\epsilon > 0$, returns a collection of vertex-disjoint cycles that covers a number of vertices that is at least $\frac{4}{k^2}(1-\epsilon)$ of the omniscient optimum using $O_{\epsilon,k}(1)$ queries per element, hence $O_{\epsilon,k}(n)$ queries overall.

To better appreciate the challenge we face, we note that even in the stochastic matching setting, we do not have a clear idea of how large the omniscient optimum is. Indeed, there is a significant body of work on the expected cardinality of matching in *complete* random graphs (see, e.g., [61, Chapter 7]), where the omniscient optimum is known to be close to n. But in our work we are dealing with *arbitrary* graphs where it can be a much smaller number. In addition, naïve algorithms fail to achieve our goal, even if they are allowed many queries. For example, querying a sublinear number of edges incident to each vertex, chosen uniformly at random, gives a vanishing fraction of the omniscient optimum—as we show in Section 11.4.

The primary technical ingredient in the design of our *adaptive algorithm* is that if, in any round r of the algorithm, the solution computed by round r (based on previous queries) is small compared to the omniscient optimum, then the current structure must admit a *large collection of disjoint constant-sized 'augmenting' structures*. These augmenting structures are composed of edges that have not been queried so far. Of course, we do not know whether these structures we are counting on to help augment our current matching actually exist; but we do know that these augmenting structures have constant size (and so each structure exists with some constant probability) and are *disjoint* (and therefore the outcomes of the queries to the different augmenting structures are independent). Hence, by querying all these structures in parallel in round r, in expectation, we can close a constant fraction of the gap between our current solution and the omniscient optimum. By repeating this argument over a constant number of rounds, we achieve a $(1-\epsilon)$ fraction of the omniscient optimum. In the case of stochastic matching, these augmenting structures are simply augmenting paths; in the more general case of k-cycle packing, we borrow the notion of augmenting structures from Hurkens and Schrijver [161].

11.1.2 Our experimental results: Application to kidney exchange

Our work is directly motivated by applications to kidney exchange. Mathematically, we can consider a directed graph $G = (V, E)$, where each node represents a donor-patient pair, an edge (u, v) means that the donor of pair u is possibly compatible with the patient of pair v. In this graph, pairwise and 3-way exchanges correspond to 2-cycles and 3-cycles, respectively. The edges of the compatibility graph E are determined by the medical characteristics—blood type and tissue type—of donors and patients. Querying an $e \in E$ refers to performing the corresponding crossmatch test. While some patients are more likely to pass crossmatch tests than others the average is as low as 30% in major kidney exchange programs [15, 107, 189]. This means that, if we only tested a perfect pairwise matching over n donor-patient pairs, we would expect only $0.09n$ of the patients to actually receive a kidney. In contrast, the omniscient solution that runs crossmatch tests on all possible pairwise exchanges (in the compatibility graph) may be able to provide kidneys to all n patients; but this solution is impractical.

Our adaptive algorithms for stochastic pairwise matching uncovers a sweet spot between these two extremes. On the one hand, it only mildly increases medical expenses, from one crossmatch

Figure 11.1: Compatibility graphs for pairwise and three-way exchanges. Solid blue edges represent successful crossmatch tests, dashed blue edges represent failed crossmatch tests, and black edges represent potential compatibilities that have not been tested. Note that when pairwise exchanges are considered, the number of incoming edge tests of a node is the same as the number of its outgoing edge tests—a patient and its willing but incompatible donor are always involved in an equal number of tests—while in three-way exchanges the number of incoming and outgoing edge tests may be different.

test per patient, to a larger, yet constant, number; and it is highly parallelizable, requiring only a constant number of rounds, so the time required to complete all crossmatch tests does not scale with the number of donors and patients. On the other hand, the adaptive algorithm essentially recovers the entire benefit of testing all potentially feasible pairwise exchanges. The qualitative message of this theoretical result is clear: *a mild increase in number of crossmatch tests provides nearly the full benefit of exhaustive testing.* When 3-way exchanges are considered, our adaptive algorithm for 3-cycle packing provides a $(4/9)$-approximation to the omniscient optimum, using only $O(1)$ crossmatch tests per patient and $O(n)$ overall. While the practical implications of this result are currently not as crisp as those of its pairwise counterpart, future work may improve the approximation ratio (using $O(n)$ queries and an exponential-time algorithm), as we explain in Section 11.9.1.

To bridge the gap between theory and practice, we provide experiments for pairwise and 3-way exchanges on both simulated data and real data from the first 169 match runs of the United Network for Organ Sharing (UNOS) US nationwide kidney exchange, which now includes 153 transplant centers—approximately 66% of the transplant centers in the US. The exchange began matching in October 2010 and now matches on a biweekly basis. Using adaptations of the algorithms presented in this chapter, we show that even a small number of non-adaptive rounds, followed by a single period during which only those edges selected during those rounds are queried, results in large gains relative to the omniscient pairwise or 3-way exchanges. We discuss the policy implications of this promising result in Section 11.9.2.

11.2 Related work

While works on stochastic matching often draw on kidney exchange for motivation—or at least mention it in passing—these two research areas are almost disjoint. We therefore discuss them separately in Sections 11.2.1 and 11.2.2.

11.2.1 Stochastic matching

Prior work has considered multiple variants of stochastic matching. A popular variant is the *query-commit* problem, where the algorithm is *forced* to add any queried edge to the matching if the edge is found to exist. Goel and Tripathi [130] establish an upper bound of 0.7916 for graphs in which no information is available about the edges, while Costello et al. [87] establish a lower bound of 0.573 and an upper bound of 0.898 for graphs in which each edge e exists with a given probability p_e. Molinaro and Ravi [209] propose an algorithm for 2-cycle matching in the query-commit model that is nearly optimal given additional theoretical assumptions. Similarly to our work, these approximation ratios are with respect to the omniscient optimum, but the informational disadvantage of the algorithm stems purely from the query-commit restriction.

Within the query-commit setting, another thread of work [4, 42, 77] imposes an additional *per-vertex budget constraint* where the algorithm is not allowed to query more than a specified number, b_v, of edges incident to vertex v. With this additional constraint, the benchmark that the algorithm is compared to switches from the omniscient optimum to the constrained optimum, i.e., the performance of the best decision tree that obeys the per-vertex budget constraints and the query-commit restriction. In other words, the algorithm's disadvantage compared to the benchmark is only that it is constrained to run in polynomial-time. Here, again, the best known approximation ratios are constant. A generalization of these results to packing problems has been studied by Gupta and Nagarajan [136].

Similarly to our work, Blum et al. [55] consider a stochastic matching setting without the query-commit constraint. They set the per-vertex budget to exactly 2, and ask which subset of edges is queried by the optimal collection of queries subject to this constraint. They prove structural results about the optimal solution, which allow them to show that finding the optimal subset of edges to query is **NP**-hard. In addition, they give a polynomial-time algorithm that finds an almost optimal solution on a class of random graphs (inspired by kidney exchange settings). Crucially, the benchmark of Blum et al. [55] is also constrained to two queries per vertex.

There is a significant body of work in stochastic optimization more broadly, for instance, the papers of Dean et al. [97] (Stochastic Knapsack), Gupta et al. [137] (Stochastic Orienteering), and Asadpour et al. [13] (Stochastic submodular maximization).

11.2.2 Kidney exchange

Early models of kidney exchange did not explicitly consider the setting where an edge that is chosen to be matched only exists probabilistically. Recent research by Dickerson et al. [107] and Anderson et al. [9] focuses on the kidney exchange application and restricts attention to a single crossmatch test per patient (the current practice), with a similar goal of maximizing the expected number of matched vertices, in a realistic setting (for example, they allow 3-cycles and chains initiated by altruistic donors, who enter the exchange without a paired patient). They develop integer programming techniques, which are empirically evaluated using real and synthetic data. As opposed to that line of work, which takes into account a single compatibility test per patient, our work considers the benefit that multiple tests per patient can bring to the quality of the matching. Manlove and OMalley [199] discuss the integer programming formulation used by the national exchange in the United Kingdom, which takes edge failures into account in an ad

hoc way by, for example, preferring shorter cycles to longer ones. To our knowledge, our work is the first to describe a general method for testing any number of edges *before* the final match run is performed—and to provide experiments on real data showing the expected effect on fielded exchanges of such edge querying policies.

Another form of stochasticity present in fielded kidney exchanges is the arrival and departure of donor-patient pairs over time (and the associated arrival and departure of their involved edges in the compatibility graph). Recent work has addressed this added form of dynamism from a theoretical [6, 8, 260] and experimental [22, 104, 106] point of view. Theoretical models have not addressed the case where an edge in the current graph may not exist (as we do in this chapter); the more recent experimental papers have incorporated this possibility, but have not considered the problem of querying edges before recommending a final matching. We leave as future research the analysis of edge querying in stochastic matching in such a dynamic model.

11.2.3 Subsequent Work

The publication of our results [58] motivated a followup work by Assadi et al. [19]. In their work, Assadi et al. [19] consider the stochastic matching (2-cycle packing) problem, and show that pre-processing the graph before applying our algorithm achieves the same approximation guarantee using fewer queries per vertex. In particular, for both our adaptive and non-adaptive algorithms, the number of queries per vertex, even though independent of the number of vertices, is exponential in $1/\epsilon$. For the particular case of 2-cycle matching, Assadi et al. [19] show that performing a vertex sparsification step before applying our algorithm obtains a similar approximation guarantee, i.e., $(1-\epsilon)$ for adaptive and $0.5(1-\epsilon)$ for the non-adaptive algorithms, using a number of queries that is polynomial in $1/\epsilon$.

11.3 The Model

For any graph $G = (V, E)$, let $M(E)$ denote its maximum (cardinality) matching. In the notation $M(E)$, we intentionally suppress the dependence on the vertex set V, since we are only interested in the maximum matchings of different subsets of edges for a fixed vertex set. In addition, for two matchings M and M', we denote their *symmetric difference* by $M \Delta M' = (M \cup M') \setminus (M \cap M')$; it includes only paths and cycles consisting of alternating edges of M and M'.

In the stochastic setting, given a set of edges X, define X_p to be the random subset formed by including each edge of X independently with probability p. We will assume for ease of exposition that $p_e = p$ for all edges $e \in E$. Our results hold when p is a lower bound, i.e., $p_e \geq p$ for all $e \in E$.

Given a graph $G = (V, E)$, define $\overline{M}(E)$ to be $\mathbb{E}[|M(E_p)|]$, where the expectation is taken over the random draw E_p. In addition, given the results of queries on some set of edges T, define $\overline{M}(E|T)$ to be $\mathbb{E}[|M(X_p \cup T')|]$, where $T' \subseteq T$ is the subset of edges of T that are known to exist based on the queries, and $X = E \setminus T$.

In the *non-adaptive* version of our problem, the goal is to design an algorithm that, given a graph $G = (V, E)$ with $|V| = n$, queries a subset X of edges in parallel such that $|X| = O(n)$, and maximizes the ratio $\overline{M}(X)/\overline{M}(E)$.

In contrast, an *adaptive* algorithm proceeds in rounds, and in each round queries a subset of edges in parallel. Based on the results of the queries up to the current round, it can choose the subset of edges to test in the next round. Formally, an R-*round adaptive* stochastic matching algorithm selects, in each round r, a subset of edges $X_r \subseteq E$, where X_r can be a function of the results of the queries on $\bigcup_{i<r} X_i$. The objective is to maximize the ratio $\mathbb{E}[|M(\bigcup_{1\leq i\leq R} X_i)|]/\overline{M}(E)$, where the expectation in the numerator is taken over the outcome of the query results and the sets X_i chosen by the algorithm.

11.4 Understanding the Challenges

To gain some intuition for our goal of arbitrarily good approximations to the omniscient optimum, and why it is challenging, let us consider a naïve algorithm and understand why it fails. This non-adaptive algorithm schedules $R = O(\log(n)/p)$ queries for each vertex as follows. First, order all vertices arbitrarily and start with an empty set of queries. In order, for each vertex v, let $N_R(v)$ be the set of neighbors of v for whom at most R queries have been scheduled. Schedule $\min\{R, N_R(v)\}$ queries, each between v and an element of $N_R(v)$, where these elements are selected uniformly at random from $N_R(v)$.

The next example shows that this proposed algorithm only achieves $\frac{5}{6}$ fraction of the omniscient optimal solution, as opposed to our goal of achieving arbitrarily good $(1-\epsilon)$ approximations to the omniscient optimal. Furthermore, in the following example when each edge exists with probability $p > \frac{5}{6}$, this algorithm still only achieves a $\frac{5}{6}$ fraction of the omniscient optimal solution, which is worse than a trivial algorithm of just picking *one* maximum matching that guarantees a matching of size pn.

Example 11.4.1. *Consider the graph $G = (V, E)$ whose vertices are partitioned into sets A, B, C, and D, such that $|A| = |B| = \frac{n}{2}$ and $|C| = |D| = n$. Let E consist of two random bipartite graphs of degree $R = O(\log(n)/p)$ between A and B and similarly between C and D. And let B and C be connected with a* complete *bipartite graph. Let p be the existence probability of any edge.*

With high probability, there is a perfect matching that matches A to B and C to D. However, by the time the algorithm has processed half of the vertices, in expectation half of the vertices in A, B, C, and D are processed. For every vertex in B, this vertex has more neighbors in C than in D. So, at this point, with high probability all of the vertices of B already have R queries scheduled from half of the vertices in C. Therefore, after this point in the algorithm, no edges between A and B will be queried. So, half of the vertices in A remain unmatched. Compared to the omniscient optimum—which is a perfect matching with high probability—the approximation ratio of this algorithm is at most $\frac{5}{6}$.

In the aforementioned example, $N_R(v)$ is restricted to vertices that have received at most R queries in order to bias the choice of queries towards vertices with fewer scheduled queries. At a high level, this is done to avoid scheduling queries for vertices that have already found existing and suitable matches. In the next example, we show that a naïve algorithm that uniformly queries $o(n)$ neighbors of each vertex—and therefore does not bias the queries towards vertices with fewer existing queries—suffers from even worse performance.

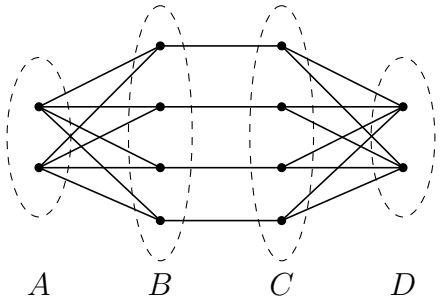


Figure 11.2: Illustration of the construction in Example 11.4.2, for $t = 4$ and $\beta = 1/2$.

Example 11.4.2. *Consider the graph $G = (V, E)$ whose vertices are partitioned into sets A, B, C, and D, such that $|A| = |D| = t^\beta$ and $|B| = |C| = t$, for some $1 > \beta > 0$. Note that in this graph $n = \Theta(t)$. Let E consist of one perfect matching between the vertices of B and C, and two complete bipartite graphs, one between A and B, and another between C and D. See Figure 11.2 for an illustration. Let $p = 0.5$ be the existence probability of any edge.*

The omniscient optimal solution can use any edge, and, in particular, it can use the edges between B and C. Since, these edges form a matching of size t and $p = 0.5$, they alone provide a matching of expected size $t/2$. Hence, $\overline{M}(E) \geq t/2$.

Now, for any $\alpha < \beta$, consider the algorithm that queries t^α random neighbors for each vertex. For every vertex in B, the probability that its edge to C is chosen is at most $\frac{t^\alpha}{t^\beta+1}$ (similarly for the edges from C to B). Therefore, the expected number of edges chosen between B and C is at most $\frac{2t^{1+\alpha}}{t^\beta+1}$, and the expected number of existing edges between B and C, after the coin tosses, is at most $\frac{t^{1+\alpha}}{t^\beta+1}$. A and D each have t^β vertices, so they contribute at most $2t^\beta$ edges to any matching. Therefore, the expected size of the overall matching is no more than $t^{1+\alpha-\beta} + 2t^\beta$. Using $n = \Theta(t)$, we conclude that the approximation ratio of the naïve algorithm approaches 0, as $n \to \infty$. For $\alpha = 0.5$ and $\beta = 0.75$, the approximation ratio of the naïve algorithm is $O(1/n^{0.25})$, at best.

11.5 Adaptive Algorithm: $(1 - \epsilon)$-approximation

In this section, we present our main result: an adaptive algorithm—formally given as Algorithm 11.1—that achieves a $(1 - \epsilon)$ approximation to the omniscient optimum for arbitrarily small $\epsilon > 0$, using $O(1)$ queries per vertex and $O(1)$ rounds.

The algorithm is initialized with the empty matching M_0. At the end of each round r, our goal is to maintain a maximum matching M_r on the set of edges that are known to exist (based on queries made so far). To this end, at round r, we compute the maximum matching O_r on the set of edges that are known to exist *and* the ones that have not been queried yet (Step 5). We consider augmenting paths in $O_r \Delta M_{r-1}$, and query all the edges in them (Steps 6 and 7). Based on the results of these queries (Q_r), we update the maximum matching (M_r). Finally, we return the maximum matching M_R computed after $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ rounds. (Let us assume that R is an integer for ease of exposition.)

It is easy to see that *this algorithm queries at most $\frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ edges per vertex*: In a given round

Algorithm 11.1: Adaptive Algorithm for Stochastic Matching: $(1-\epsilon)$ approximation

Input: A graph $G = (V, E)$, parameters ϵ, and p.
Let $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$
Initialize M_0 to the empty matching and $W_1 \leftarrow \emptyset$.
for $r = 1, \ldots, R$ **do**
    Compute a maximum matching, O_r, in $(V, E \setminus W_r)$.
    Set Q_r to the collection of all augmenting paths of M_{r-1} in $O_r \Delta M_{r-1}$.
    Query the edges in Q_r. Let Q'_r and Q''_r be the set of existing and non-existing edges.
    $W_{r+1} \leftarrow W_r \cup Q''_r$.
    Set M_r to the maximum matching in $\left(V, \bigcup_{j=1}^{r} Q'_j\right)$.
end for
return M_R.

r, the algorithm queries edges that are in augmenting paths of $O_r \Delta M_{r-1}$. Since there is at most one augmenting path using any particular vertex, the algorithm queries at most one edge per vertex in each round. Furthermore, the algorithm executes $\frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ rounds. Therefore, the number of queries issued by the algorithm per vertex is as claimed.

The rest of the section is devoted to proving that the matching returned by this algorithm after R rounds has cardinality that is, in expectation, at least a $(1-\epsilon)$ fraction of $\overline{M}(E)$.

Theorem 11.5.1. *For any graph $G = (V, E)$ and any $\epsilon > 0$, Algorithm 11.1 returns a matching whose expected cardinality is at least $(1-\epsilon)\ \overline{M}(E)$ in $R = \frac{\log(2/\epsilon)}{p^{(2/\epsilon)}}$ rounds.*

As mentioned in Section 11.1, one of the insights behind this result is the existence of many *disjoint* augmenting paths of *bounded length* that can be used to augment a matching that is far from the omniscient optimum, that is, a lower bound on the number of elements in Q_r of a given length L. This observation is formalized in the following lemma. (We emphasize that the lemma pertains to the non-stochastic setting.)

Lemma 11.5.2. *Consider a graph $G = (V, E)$ with two matchings M_1 and M_2. Suppose $|M_2| > |M_1|$. Then in $M_1 \Delta M_2$, for any odd length $L \geq 1$, there exist at least $|M_2| - (1 + \frac{2}{L+1})|M_1|$ augmenting paths of length at most L, which augment the cardinality of M_1.*

Proof. Let x_l be the number of augmenting paths of length l (for any odd $l \geq 1$) found in $M_1 \Delta M_2$ that augment the cardinality of M_1. Each augmenting path increases the size of M_1 by 1, so the total number of augmenting paths $\sum_{l \geq 1} x_l$ is at least $|M_2| - |M_1|$. Moreover, each augmenting path of length l has $\frac{l-1}{2}$ edges in M_1. Hence, $\sum_{l \geq 1} \frac{l-1}{2} x_l \leq |M_1|$. In particular, this implies that $\frac{L+1}{2} \sum_{l \geq L+2} x_l \leq |M_1|$. We conclude that

$$\sum_{l=1}^{L} x_l = \sum_{l \geq 1} x_l - \sum_{l \geq L+2} x_l \geq (|M_2| - |M_1|) - \frac{2}{L+1}|M_1| = |M_2| - \left(1 + \frac{2}{L+1}\right)|M_1|.$$

□

The rest of the theorem's proof requires some additional notation. At the beginning of any given round r, the algorithm already knows about the existence (or non-existence) of the edges in $\bigcup_{i=1}^{r-1} Q_i$. We use Z_r to denote the expected size of the maximum matching in graph $G = (V, E)$ given the results of the queries $\bigcup_{i=1}^{r-1} Q_i$. More formally, $Z_r = \overline{M}(E|\bigcup_{i=1}^{r-1} Q_i)$. Note that $Z_1 = \overline{M}(E)$.

For a given r, we use the notation $\mathbb{E}_{Q_r}[X]$ to denote the expected value of X where the expectation is taken *only* over the outcome of query Q_r, and fixing the outcomes on the results of queries $\bigcup_{i=1}^{r-1} Q_i$. Moreover, for a given r, we use $\mathbb{E}_{Q_r,\ldots,Q_R}[X]$ to denote the expected value of X with the expectation taken over the outcomes of queries $\bigcup_{i=r}^{R} Q_i$, and fixing an outcome on the results of queries $\bigcup_{i=1}^{r-1} Q_i$.

In Lemma 11.5.3, for any round r and for *any* outcome of the queries $\bigcup_{i=1}^{r-1} Q_i$, we lower-bound the *expected increase in the size of* M_r over the size of M_{r-1}, with the expectation being taken only over the outcome of edges in Q_r. This lower bound is a function of Z_r.

Lemma 11.5.3. *For any $r \in [R]$, odd L, and $Q_1, \cdots, Q_{r-1}$, it holds that $\mathbb{E}_{Q_r}[|M_r|] \geq (1-\gamma)|M_{r-1}| + \alpha Z_r$, where $\gamma = p^{(L+1)/2}\left(1+\frac{2}{L+1}\right)$ and $\alpha = p^{(L+1)/2}$.*

Proof. By Lemma 11.5.2, there exist at least $|O_r| - (1+\frac{2}{L+1})|M_{r-1}|$ augmenting paths in $O_r \Delta M_{r-1}$ that augment M_{r-1} and are of length at most L. The O_r part of every augmenting path of length at most L exists independently with probability at least $p^{(L+1)/2}$. Therefore, the expected increase in the size of the matching is:

$$\begin{aligned}\mathop{\mathbb{E}}_{Q_r}[|M_r|] - |M_{r-1}| \;&\geq\; p^{\frac{L+1}{2}}\left(|O_r| - \left(1+\frac{2}{L+1}\right)|M_{r-1}|\right)\\ &=\; \alpha|O_r| - \gamma|M_{r-1}| \;\geq\; \alpha Z_r - \gamma|M_{r-1}|,\end{aligned}$$

where the last inequality holds by the fact that Z_r, which is the expected size of the optimal matching with expectation taken over non-queried edges, cannot be larger than O_r, which is the maximum matching assuming that every non-queried edge exists. □

We are now ready to prove the theorem.

Proof of Theorem 11.5.1. Let $L = \frac{4}{\epsilon} - 1$; it is assumed to be an odd integer for ease of exposition. Otherwise there exists $\epsilon/2 \leq \epsilon' \leq \epsilon$ such that $\frac{4}{\epsilon'} - 1$ is an odd integer. We use a similar simplification in the proofs of other results. By Lemma 11.5.3, we know that for every $r \in [R]$, $\mathbb{E}_{Q_r}[|M_r|| \geq (1-\gamma)|M_{r-1}| + \alpha Z_r$, where $\gamma = p^{(L+1)/2}(1+\frac{2}{L+1})$, and $\alpha = p^{(L+1)/2}$. We will use this inequality repeatedly to derive our result. We will also require the equality

$$\mathop{\mathbb{E}}_{Q_{r-1}}[Z_r] = \mathop{\mathbb{E}}_{Q_{r-1}}\left[\overline{M}(E|\bigcup_{i=1}^{r-1} Q_i)\right] = \overline{M}(E|\bigcup_{i=1}^{r-2} Q_i) = Z_{r-1}. \tag{11.1}$$

First, applying Lemma 11.5.3 at round R, we have that $\mathbb{E}_{Q_R}[|M_R|] \geq (1-\gamma)|M_{R-1}| + \alpha Z_R$. This inequality is true for any fixed outcomes of $Q_1, \ldots, Q_{R-1}$. In particular, we can take the expectation over Q_{R-1}, and obtain

$$\mathop{\mathbb{E}}_{Q_{R-1},Q_R}[|M_R|] \geq (1-\gamma) \mathop{\mathbb{E}}_{Q_{R-1}}[|M_{R-1}|] + \alpha \mathop{\mathbb{E}}_{Q_{R-1}}[Z_R].$$

By Equation (11.1), we know that $\mathbb{E}_{Q_{R-1}}[Z_R] = Z_{R-1}$. Furthermore, we can apply Lemma 11.5.3 to $\mathbb{E}_{Q_{R-1}}[|M_{R-1}|]$ to get the following inequality:

$$\begin{aligned}\mathop{\mathbb{E}}_{Q_{R-1},Q_R}[|M_R|] &\geq (1-\gamma) \mathop{\mathbb{E}}_{Q_{R-1}}[|M_{R-1}|] + \alpha \mathop{\mathbb{E}}_{Q_{R-1}}[Z_R] \\ &\geq (1-\gamma)\ ((1-\gamma)\,|M_{R-2}| + \alpha\, Z_{R-1}) + \alpha\, Z_{R-1} \\ &= (1-\gamma)^2\,|M_{R-2}| + \alpha\,(1+(1-\gamma))\,Z_{R-1}.\end{aligned}$$

We repeat the above steps by sequentially taking expectations over Q_{R-2} through Q_1, and at each step applying Equation (11.1) and Lemma 11.5.3. This gives us

$$\begin{aligned}\mathop{\mathbb{E}}_{Q_1,\ldots,Q_R}[|M_R|] \;&\geq\; (1-\gamma)^R|M_0| + \alpha\,(1+(1-\gamma)+\cdots+(1-\gamma)^{R-1})Z_1 \\ &=\; \alpha\,\frac{1-(1-\gamma)^R}{\gamma}Z_1,\end{aligned}$$

where the second transition follows from the initialization of M_0 as an empty matching. Since $L = \frac{4}{\epsilon} - 1$ and $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$, we have

$$\frac{\alpha}{\gamma}\big(1-(1-\gamma)^R\big) = \left(1-\frac{2}{L+1}\right)\big(1-(1-\gamma)^R\big) \geq 1-\frac{2}{L+1}-e^{-\gamma R} \geq 1-\frac{\epsilon}{2}-\frac{\epsilon}{2} = 1-\epsilon, \tag{11.2}$$

where the second transition is true because $e^{-x} \geq 1-x$ for all $x \in \mathbb{R}$. We conclude that $\mathbb{E}_{Q_1,\ldots,Q_R}[|M_R|] \geq (1-\epsilon)\,Z_1$. Because $Z_1 = \overline{M}(E)$, it follows that the expected size of the algorithm's output is at least $(1-\epsilon)\,\overline{M}(E)$. □

11.6 Non-adaptive algorithm: 0.5-approximation

The adaptive algorithm, Algorithm 11.1, augments the current matching by computing a maximum matching on queried edges that are known to exist, and edges that have not been queried. One way to extend this idea to the non-adaptive setting is the following: we can simply choose several edge-disjoint matchings, and hope that they help in augmenting each other. In this section, we ask: How close can this non-adaptive interpretation of our adaptive approach take us to the omniscient optimum?

In more detail, our non-adaptive algorithm—formally given as Algorithm 11.2—iterates $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ times. In each iteration, it picks a maximum matching and removes it. The set of edges queried by the algorithm is the union of the edges chosen in some iteration. We will show that, for any arbitrarily small $\epsilon > 0$, the algorithm finds a $0.5(1-\epsilon)$-approximate solution. Since we allow an arbitrarily small (though constant) probability p for stochastic matching, achieving a 0.5-approximation independently of the value of p, while querying only a linear number of edges, is nontrivial. For example, a naïve algorithm that only queries one maximum matching clearly does not guarantee a 0.5-approximation—it would guarantee only a p-approximation. In addition, the example given in Section 11.3 shows that choosing edges at random performs poorly.

Algorithm 11.2: Non-adaptive algorithm for Stochastic Matching: 0.5-approximation

Input: A graph $G(V, E)$, parameters ϵ and p.
Let $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$
Initialize $W_0 \leftarrow \emptyset$.
for $r = 1, \ldots, R$ **do**
    Compute a maximum matching, O_r, in $(V, E \setminus W_{r-1})$.
    $W_r \leftarrow W_{r-1} \cup O_r$.
end for
Query all the edges in W_R, and output the maximum matching among the edges that are found to exist in W_R.

The number of edges incident to any particular vertex that are queried by the algorithm is at most $\frac{\log(2/\epsilon)}{p^{2/\epsilon}}$, because the vertex can be matched with at most one neighbor in each round. The next theorem establishes the approximation guarantee of Algorithm 11.2.

Theorem 11.6.1. *Given a graph $G = (V, E)$ and any $\epsilon > 0$, the expected size $\overline{M}(W_R)$ of the matching produced by Algorithm 11.2 is at least a $0.5(1 - \epsilon)$ fraction of $\overline{M}(E)$.*

Similar to the adaptive procedure of Section 11.5, the proof of Theorem 11.6.1 relies on analyzing how much matching O_r increases the size of the expected matching, $\overline{M}(W_{r-1})$, at every round. As opposed to the adaptive procedure, here we do not query the edges in W_{r-1}. Hence, we need to reason about the *expected size* of the matching up to round r, $\overline{M}(W_{r-1})$, and the expected size of the matching in the remaining graph, $\overline{M}(E \setminus W_{r-1})$. The following Lemma can be used to bound $\overline{M}(E \setminus W_{r-1})$ in terms of $\overline{M}(W_{r-1})$ and $\overline{M}(E)$.

Lemma 11.6.2. *Let E_1 be an arbitrary subset of edges of E, and let $E_2 = E \setminus E_1$. Then $\overline{M}(E) \leq \overline{M}(E_1) + \overline{M}(E_2)$.*

Proof. Let E' be an arbitrary subset of edges of E, and let $E'_1 = E_1 \cap E'$ and $E'_2 = E_2 \cap E'$. We claim that $|M(E')| \leq |M(E'_1)| + |M(E'_2)|$. This is because if T is the set of edges in a maximum matching in graph (V, E'), then clearly $T \cap E'_1$ and $T \cap E'_2$ are valid matchings in E'_1 and E'_2 respectively, and thereby it follows that $|M(E'_1)| \geq |T \cap E'_1|$ and $|M(E'_2)| \geq |T \cap E'_2|$, and hence $|M(E')| \leq |M(E'_1)| + |M(E'_2)|$. Expectation is a convex combination of the values of the outcomes. For every subset E' of edges in E, multiplying the above inequality by the probability that the outcome of the coin tosses on the edges of E is E', and then summing the various inequalities, we get $\overline{M}(E) \leq \overline{M}(E_1) + \overline{M}(E_2)$. □

In order to lower bound $\overline{M}(W_R)$, we first show that for any round r, either our current collection of edges has an expected matching size $\overline{M}(W_{r-1})$ that compares well with $\overline{M}(E)$, or in round r, we have a significant increase in $\overline{M}(W_r)$ over $\overline{M}(W_{r-1})$.

Lemma 11.6.3. *At any iteration $r \in [R]$ of Algorithm 11.2 and odd L, if $\overline{M}(W_{r-1}) \leq \overline{M}(E)/2$, then*

$$\overline{M}(W_r) \geq \frac{\alpha}{2}\,\overline{M}(E) + (1 - \gamma)\overline{M}(W_{r-1}),$$

where $\gamma = p^{(L+1)/2}(1 + \frac{2}{L+1})$ and $\alpha = p^{(L+1)/2}$.

Proof. Assume that $\overline{M}(W_{r-1}) \leq \overline{M}(E)/2$. By Lemma 11.6.2, we know that $\overline{M}(E \setminus W_{r-1}) \geq \overline{M}(E) - \overline{M}(W_{r-1})$. Recall that O_r is the maximum matching left in $E \setminus W_{r-1}$, therefore, $|O_r| = |M(E \setminus W_{r-1})| \geq \overline{M}(E \setminus W_{r-1}) \geq \overline{M}(E) - \overline{M}(W_{r-1}) \geq \overline{M}(E)/2$.

In a thought experiment, say at the beginning of round r, we query the set W_{r-1} and let W'_{r-1} be the set of edges that are found to exist. By Lemma 11.5.2, there are at least $|O_r| - (1 + \frac{2}{L+1})|M(W'_{r-1})|$ augmenting paths of length at most L in $O_r \Delta M(W'_{r-1})$ that augment $M(W'_{r-1})$. Each of these paths succeeds with probability at least $p^{(L+1)/2}$. We have,

$$\overline{M}(O_r \cup W'_{r-1}|W'_{r-1}) - |M(W'_{r-1})| \geq p^{(L+1)/2} \left(|O_r| - (1 + \frac{2}{L+1})|M(W'_{r-1})| \right) \tag{11.3}$$

$$\geq p^{(L+1)/2} \left(\frac{1}{2}\overline{M}(E) - (1 + \frac{2}{L+1})|M(W'_{r-1})| \right), \tag{11.4}$$

where the expectation on the left hand side is taken only over the outcome of the edges in O_r. Therefore, we have $\overline{M}(O_r \cup W'_{r-1}|W'_{r-1}) \geq \frac{\alpha}{2}\overline{M}(E) + (1-\gamma)|M(W'_{r-1})|$, where $\alpha = p^{(L+1)/2}$ and $\gamma = p^{(L+1)/2}\,(1 + \frac{2}{L+1})$. Taking expectation over the coin tosses on W_{r-1} that create outcome W'_{r-1}, we have our result, i.e.,

$$\overline{M}(W_r) \geq \mathop{\mathbb{E}}_{W_{r-1}}[\overline{M}(O_r \cup W'_{r-1}|W'_{r-1})] \geq \overline{M}(O_r \cup W_{r-1}) \geq \frac{\alpha}{2}\overline{M}(E) + (1-\gamma)\overline{M}(W_{r-1}).$$

□

We are now ready to prove Theorem 11.6.1.

Proof of Theorem 11.6.1. For ease of exposition, assume $L = \frac{4}{\epsilon} - 1$ is an odd integer. Then, either $\overline{M}(W_R) \geq \overline{M}(E)/2$ in which case we are done. Or otherwise, by repeatedly applying Lemma 11.6.3 for R steps, we have

$$\overline{M}(W_R) \;\geq\; \frac{\alpha}{2}(1 + (1-\gamma) + (1-\gamma)^2 + \cdots + (1-\gamma)^{R-1})\overline{M}(E) \;\geq\; \frac{\alpha}{2}\frac{(1-(1-\gamma)^R)}{\gamma}\overline{M}(E).$$

Now, $\frac{\alpha}{\gamma}(1-(1-\gamma)^R) \geq 1 - \frac{2}{L+1} - e^{-\gamma R} \geq 1 - \epsilon$ for $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$. Hence, we have our $0.5(1-\epsilon)$ approximation. □

11.6.1 Upper Bound on the Performance of the Non-Adaptive Algorithm

As we explain in more details in Section 11.9.1, we do not know whether in general non-adaptive algorithms can achieve a $(1-\epsilon)$-approximation with $O_\epsilon(1)$ queries per vertex. However, if there is such an algorithm, it is not Algorithm 11.2! Indeed, the next theorem shows that the algorithm cannot give an approximation ratio better than $11/12$ to the omniscient optimum. This fact holds even when $R = \Theta(\log n)$.

Theorem 11.6.4. *Let $p = 0.5$. For any $\epsilon > 0$ there exists n and a graph (V, E) with $|V| \geq n$ such that Algorithm 11.2, with $R = O(\log n)$, returns a matching with expected size of at most $\frac{11}{12}\overline{M}(E) + \epsilon$.*

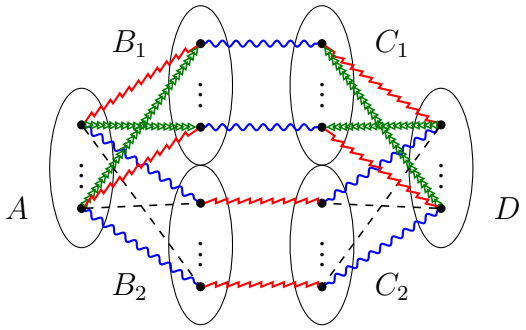


Figure 11.3: Illustration of the upper bound on the performance of non-adaptive algorithm. Blue and red edges represent the matching picked at rounds 1 and 2, respectively. The green edges represent the edges picked at round 3 and above. The dashed edges are never picked by the algorithm.

Before proving Theorem 11.6.4, let us first show that the expected size of a matching, i.e., $\overline{M}(E)$, is large in a complete bipartite graph.

Lemma 11.6.5. *Let $G = (U \cup V, U \times V)$ be a complete bipartite graph between U and V with $|U| = |V| = n$. For any constant probability p, $\overline{M}(E) \geq n - o(n)$.*

Proof. Denote by E_p the random set of edges formed by including each edge in $U \times V$ independently with probability p. We show that with probability at least $1 - \frac{1}{n^8}$, over the draw E_p, the maximum matching in the graph $(U \cup V, E_p)$ is at least $n - c\log(n)$, where $c = 10/\log(\frac{1}{1-p})$, and this will complete our claim.

In order to show this, we prove that with probability at least $1 - \frac{1}{n^8}$, over the draw E_p, all subsets $S \subseteq U$ of size at most $n - c\log(n)$, have a neighborhood of size at least $|S|$. By Hall's theorem, our claim will follow.

Consider any set $S \subseteq U$ of size at most $n - c\log(n)$. We will call set S 'bad' if there exists some set $T \subseteq V$ of size $(|S| - 1)$ such that S does not have edges to $V \setminus T$. Fix any set $T \subseteq V$ of size $|S| - 1$. Over draws of E_p, the probability that S has no outgoing edges to $V \setminus T$ is at most $(1-p)^{|S||V\setminus T|} = (1-p)^{|S|(n-|S|+1)}$. Hence, by union bound, the probability that S is bad is at most $\binom{n}{|S|-1}(1-p)^{|S|(n-|S|+1)}$.

Again, by union bound, the probability that some set $S \subseteq U$ of size at most $n - c\log(n)$ is bad is at most $\sum_{1\leq|S|\leq n-c\log(n)} \binom{n}{|S|}\binom{n}{|S|-1}(1-p)^{|S|(n-|S|+1)}$ and this in turn is at most

$$\sum_{1\leq|S|\leq n-c\log(n)} n^{|S|}n^{|S|}(1-p)^{|S|(n-|S|+1)} \leq \sum_{1\leq|S|\leq n-c\log(n)} e^{|S|\cdot(2\log(n)+(n+1)\log(1-p)-|S|\log(1-p))}$$

Note that the exponent in the summation achieves its maximum for $|S| = 1$. For $c = 10/\log(\frac{1}{1-p})$, we have that the given sum is at most $\exp(-\frac{n}{2}\log(\frac{1}{1-p}))$, and hence with high probability, no set $S \subseteq U$ of size at most $n - c\log(n)$ is bad. □

Proof of Theorem 11.6.4. Let (V, E) be a graph, illustrated in Figure 11.3, whose vertices are partitioned into sets A, B, C, and D, such that $|A| = |D| = \frac{t}{2}$, $|B| = |C| = t$. The edge set E

consists of one perfect matching between vertices of B and C, and two complete bipartite graphs, one between A and B, and another between C and D. Let $p = 0.5$ be the existence probability of any edge.

We first examine the value of the omniscient optimal, $\overline{M}(E)$. Since $p = 0.5$, in expectation, half of the edges in the perfect matching between B and C exist, and therefore half of the vertices of B and C will get matched. As we showed in Lemma 11.6.5, with high probability, the complete bipartite graph between the remaining half of B and A has a matching of size at least $t/2 - o(t)$. And similarly, with high probability, the complete bipartite graph between remaining half of C and D has a matching of size at least $t/2 - o(t)$. Therefore, $\overline{M}(E)$ is at least $\frac{3}{2}t - o(t)$.

Next, we look at Algorithm 11.2. For ease of exposition, let B_1 and B_2 denote the top and bottom half of the vertices in B. Similarly, define C_1 and C_2. Since Algorithm 11.2 picks maximum matchings arbitrarily, we show that there exists a way of picking maximum matchings such that the expected matching size of the union of the edges picked in the matching is at most $\frac{11}{8}t$ $(= \frac{11}{12}\frac{3}{2}t)$.

Consider the following choice of maximum matching picked by the algorithm: In the first round, the algorithm picks the perfect matching between B_1 and C_1, and a perfect matching between A and B_2, and a perfect matching between C_2 and D. In the second round, the algorithm picks the perfect matching between B_2 and C_2, and a perfect matching each between A and B_1, and between C_1 and D. After these two rounds, we can see that there are no more edges left between B and C. For the subsequent $R - 2$ rounds, in each round, the algorithms picks a perfect matching between A and B_1, and a perfect matching between C_1 and D. It is easy to verify that in every round, the algorithm has picked a maximum matching from the remnant graph.

We analyze the expected size of matching output by the algorithm. For each of the vertices in B_2 and C_2, the algorithm has picked only two incident edges. For any vertex in B_2 and C_2, with probability at least $(1-p)^2 = \frac{1}{4}$, none of these two incident edges exist. Hence, the expected number of vertices that are *unmatched* in B_2 and C_2 is at least $\frac{1}{4}(\frac{t}{2} + \frac{t}{2}) = \frac{t}{4}$. Hence, the total number of edges included in the matching is at most $\frac{1}{2}(3t - t/4) = \frac{11}{8}t$. This completes our claim. □

Despite this somewhat negative result, in Section 11.8, we show experimentally on realistic kidney exchange compatibility graphs that Algorithm 11.2 performs very well for even very small values of R, across a wide range of values of p.

11.7 Generalization to stochastic k-cycle packing

So far we have focused on stochastic matching, where the goal is equivalent to finding the largest 2-cycle packing. In this section, we generalize our approach to the case of k-cycle packing for any $k \geq 2$.

Formally, for a directed graph $G = (V, E)$, the corresponding k-cycle packing instance (V, A) consists of the set of vertices V and the collection $A \subseteq V^{\leq k}$ of vertices that form a directed cycle of length at most k in G. Given graph G and its corresponding k-cycle packing instance (V, A), a feasible solution to the k-cycle packing instance is a collection $B \subseteq A$ such

that the cycles in B are vertex-disjoint. Let $V(A) \subseteq V$ denote the largest set of vertices that can be covered by a feasible k-cycle packing $B \subseteq A$, i.e., vertices in $\bigcup_{c \in B} c$. Moreover, let $K(A)$ denote the feasible k-cycle packing B with largest $|B|$.

In the stochastic variant of k-cycle packing, given a graph $G = (V, E)$, we represent by $E_p \sim E$ a random subset of edges where each edge in E is included in E_p with probability p independently. We represent by $(V, A(E_p))$ the k-cycle packing instance that corresponds to the graph (V, E_p). Note that for any E_p, $A(E_p) \subseteq A$ is the set of those cycles in A whose edges appear in E_p. We denote by $\overline{V}(A) = \mathbb{E}_{E_p \sim E}\left[|V(A(E_p))|\right]$ and $\overline{K}(A) = \mathbb{E}_{E_p \sim E}\left[|K(A(E_p))|\right]$, respectively, the expected maximum number of vertices covered in a k-cycle packing, and the expected maximum cardinality of a k-cycle packing.

Note that our goal in kidney exchange is to match the largest number of donor-patient pairs, therefore, our omniscient optimum benchmark is $\overline{V}(A)$. However, a k-cycle packing B such that $|B| \geq \alpha \overline{K}(A)$ covers a number of vertices that is at least $\frac{2}{k}\alpha \overline{V}(A)$, because every cycle in B covers at least 2 vertices and the cycles in A cover at most k vertices each. Therefore, for the majority of this section, we focus on finding a k-cycle packing whose expected size is a good approximation of $\overline{K}(A)$ and as a result the number of vertices covered by it is a good approximation of $\overline{V}(A)$. We present a polynomial-time adaptive algorithm, Algorithm 11.4, that obtains a $(1-\epsilon)\frac{2}{k}$-approximation of $\overline{K}(A)$ and $(1-\epsilon)\frac{4}{k^2}$-approximation of $\overline{V}(A)$.

Theorem 11.7.1. *There exists an adaptive polynomial-time algorithm that, given a graph $G = (V, E)$, its corresponding k-cycle packing instance (V, A), and $\epsilon > 0$, uses $R = O_{\epsilon,k}(1)$ rounds and $O_{\epsilon,k}(n)$ edge queries overall, and returns a cycle-packing B_R, such that $|B_R| \geq (1-\epsilon)\frac{2}{k}\overline{K}(A)$. Moreover, $\sum_{c \in B_R} |c| \geq (1-\epsilon)\frac{4}{k^2}\overline{V}(A)$.*

Importantly, the statement of Theorem 11.5.1 for adaptive stochastic matching is a special case of the statement of Theorem 11.7.1 for $k = 2$. By contrast, we leave the case of non-adaptive algorithms for k-cycle packing for general $k \geq 2$ as an open problem—despite having presented Theorem 11.6.1 for the special case of $k = 2$—and describe some of the challenges one may face in obtaining such a general result for $k > 2$ in Section 11.9.1.

11.7.1 Augmenting structures for k-cycle packing

Finding an optimal solution to the k-cycle packing problem is **NP**-hard [3]. On the other hand, multiple approximation algorithms are known for k-cycle packing and its generalization to k-set packing, where A includes arbitrary subsets of $\leq k$ elements of V. One such algorithm is a local search algorithm of Hurkens and Schrijver [161] that uses a notion of *augmenting structures*. Given a k-cycle packing instance (V, A) and a feasible packing (one with disjoint cycles) $B \subseteq A$, (C, D) is said to be an *augmenting structure* for B if removing D and adding C to B increases its cardinality and maintains the feasibility of the packing. That is, if $(B \cup C) \setminus D$ is a collection of vertex-disjoint k-cycles and $|(B \cup C) \setminus D| > |B|$, where $C \subseteq A$ and $D \subseteq B$.

Hurkens and Schrijver [161] show that for any η there is a polynomial time (assuming η and k are constants) approximation algorithm that repeatedly augments a feasible solution using augmenting structures of size $s_{\eta,k}$, a constant that depends on k and η, and obtains a packing of cardinality $\geq (\frac{2}{k} - \eta)K(A)$. Hurkens and Schrijver [161] also show that an approximation

ratio better than $2/k$ cannot be achieved with local search of structures of constant size. The following theorem summarizes the results of Hurkens and Schrijver [161].

Lemma 11.7.2 ([161]). *Given a k-cycle packing instance (V, A) and a feasible packing B such that $|B| < \left(\frac{2}{k} - \eta\right)|K(A)|$, there exists an augmenting structure (C, D) for B such that $|C| \leq |s_{\eta,k}|$ and $|D| \leq s_{\eta,k}$ for some constant $s_{\eta,k}$ that depends only on η and k.*

Similarly to the case of 2-cycle matchings, we require *many* augmenting structures, since some may fail to exist when edges appear at random. In the following lemma, we show how to find a large number of small augmenting structures.

Lemma 11.7.3. *Given a k-cycle packing instance (V, A) and a feasible packing B such that $|B| < \left(\frac{2}{k} - \eta\right)|K(A)|$, there are $T = \frac{1}{ks_{\eta,k}}\left(|K(A)| - \frac{|B|}{\frac{2}{k}-\eta}\right)$ augmenting structures $(C_1, D_1), \ldots, (C_T, D_T)$ such that $|C_t| \leq s_{\eta,k}$ and $|D_t| \leq s_{\eta,k}$ for all $t \in [T]$, and the set of cycles appearing in C_ts are vertex-disjoint, i.e., for all $t, t' \in [T]$ such that $t \neq t'$, for any $c \in C_t$ and $c' \in C_{t'}$, we have $c \cap c' = \emptyset$. Moreover, this collection of augmenting structures can be found in polynomial time, assuming k and η to be constants.*

Algorithm 11.3: Finding constant-size disjoint augmenting structures for k-cycles

1: **input**: k-cycle packing instance (V, A) and a collection $B \subseteq A$ of disjoint sets, and parameter $s_{\eta,k}$.
2: Initialize $A_1 \leftarrow A$ and $Q \leftarrow \emptyset$.
3: **for** $t = 1, \cdots, |A|$ **do**
4: Find an augmenting structure (C_t, D_t) of size $s_{\eta,k}$ for B on the k-cycle packing instance (V, A_t).
5: Add (C_t, D_t) to Q. (If C_t is an empty set, break out of the loop.
6: Let $A_{t+1} \leftarrow A_t \setminus \{c | \exists c' \in C_t,$ such that $c \cap c' \neq \emptyset\}$.
7: **end forreturn** Q.

Proof. We prove this lemma using Algorithm 11.3. Note that in Step 6 of this algorithm all cycles that share any vertex with a cycle that is already in Q are removed. Therefore, by design the collection Q of augmenting structures returned by the algorithm satisfies the property that for any two augmenting structures $C_t, C_{t'}$ and any two cycles $c \in C_t$ and $c' \in C_{t'}$, $c \cap c' = \emptyset$. It remains to show that $|Q| \geq \frac{1}{ks_{\eta,k}}(|K(A)| - \frac{|B|}{\frac{2}{k}-\eta})$. That is, in the first $T = \frac{1}{ks_{\eta,k}}\left(|K(A)| - \frac{|B|}{\frac{2}{k}-\eta}\right)$ iterations of Step 4, we are able to find a nonempty augmenting structure for B. Using Lemma 11.7.2, it is sufficient to show that $K(A_{t+1}) \geq |B|/(\frac{2}{k} - \eta)$ for all $t \leq T$.

Note that for all t, $|C_t| \leq s_{\eta,k}$ and for each $c \in C_t$, $|c| \leq k$. Therefore, by time $t+1$, there are at most $t \cdot k \cdot s_{\eta,k}$ vertices of V that are covered by cycles that appear in $C_1, \ldots, C_t$. At most $t \cdot k \cdot s_{\eta,k}$ cycles in the largest cardinality packing of A may have one or more of these $t \cdot k \cdot s_{\eta,k}$ vertices. Note that removing these cycles from the largest cardinality packing $K(A)$ yields a packing for A_{t+1}, so, we have that

$$K(A_{t+1}) \geq K(A) - t \cdot k \cdot s_{\eta,k} \geq \frac{|B|}{\frac{2}{k} - \eta},$$

for all $t \leq T$. Using Lemma 11.7.2 completes the proof. □

11.7.2 Adaptive algorithm for k-set packing

We use the following polynomial-time algorithm for stochastic k-cycle packing to prove Theorem 11.7.1. In each round r, the algorithm maintains a feasible k-cycle packing B_r based on the k-cycles that have been queried so far. It then computes a collection Q_r of vertex-disjoint, small augmenting structures with respect to the current solution B_r (as in Lemma 11.7.3), where the augmenting structures are composed of cycles which may have unqueried edges. It then queries all edges that appear in some cycle in these augmenting structures, and uses those that are found to exist to augment the current solution and removes all cycles with a failed edge from consideration for the future rounds. The augmented solution is fed into the next round and the process is repeated.

Algorithm 11.4: Adaptive Algorithm for Stochastic k-cycle packing

Input: Graph $G = (V, E)$, $k \geq 2$, the corresponding k-cycle packing instance (V, A), and parameters ϵ, and p.
Let $\eta = \frac{\epsilon}{k}$ and $R = \frac{(\frac{2}{k}-\eta)k \cdot s_{\eta,k}}{p^{k \cdot s_{\eta,k}}} \log(\frac{2}{\epsilon})$ (For a $(1-\epsilon)(\frac{2}{k})$-approximation to $\overline{K}(A)$)
Initialize $r \leftarrow 1$, $B_1 \leftarrow \emptyset$ and $A_1 \leftarrow A$.
for $r = 1, \ldots, R$ **do**
    Let Q_r be the set of augmenting structures given by Algorithm 11.3 on the input of the k-cycle packing instance of (V, A_r), the collection B_r, and the parameter $s_{\eta,k}$.
    for each augmenting structure $(C, D) \in Q_r$. **do**
        for all cycles $c \in C$, **do** query all edges **end for**
        if for all $c \in C$ all edges of c exist, **then** $B_{r+1} \leftarrow (B_r \setminus D) \cup C$ **end if**
    end for
    $A_{r+1} \leftarrow A_r \setminus \{c \mid$ One or more edges in cycle c failed to exist$\}$.
end for
return B_R.

Similar to our matching results, for any element $v \in V$, the number of cycles in B_R that v belongs to and are queried is at most R. Indeed, in each of the R rounds, Algorithm 11.4 issues queries to vertex-disjoint augmenting structures—vertex disjoint set of cycles—and each such structure includes at most one cycle that uses vertex v.

Let us first introduce some notation that is helpful in proving Theorem 11.4. For a given r, we use the notation $\mathbb{E}_{Q_r}[X]$ to denote $\mathbb{E}_{Q_r}[X \mid Q_1, \ldots, Q_{r-1}]$, i.e., the expected value of X where the expectation is taken over the outcomes of Q_r when outcomes of $Q_1, \ldots, Q_{r-1}$ are fixed based on queries in the first $r-1$ rounds. Similarly, we use the notation $\mathbb{E}_{Q_r,\ldots,Q_R}[X]$ to denote $\mathbb{E}_{Q_r,\ldots,Q_R}[X \mid Q_1, \ldots, Q_{r-1}]$. Moreover, we denote by $\overline{K}(A \mid Q_1, \ldots, Q_{r-1})$ the expected size of the largest cardinality cycle-packing for (V, A) given the result of queries in $Q_1, \ldots, Q_{r-1}$.

Lemma 11.7.4. *For every $r \in [R]$, outcome of queries $Q_1, \ldots, Q_{r-1}$, and the corresponding*

B_{r-1}, *we have*

$$\mathop{\mathbb{E}}_{Q_r}[|B_r|] \geq (1-\gamma)|B_{r-1}| + \gamma(\frac{2}{k} - \eta)\overline{K}(A \mid Q_1, \ldots, Q_{r-1}),$$

where $\gamma = \frac{p^{k \cdot s_{\eta,k}}}{(\frac{2}{k} - \eta) \cdot k \cdot s_{\eta,k}}$.

Proof. By Lemma 11.7.3, Q_r is a collection of at least $\frac{1}{k \cdot s_{\eta,k}} \left(K(A_r) - \frac{|B_{r-1}|}{\frac{2}{k} - \eta} \right)$ augmenting structures of size at most $s_{\eta,k}$ whose cycles are all mutually vertex-disjoint. Note that at Step 10 of round $1, \ldots, r-1$, we remove any cycles that had an edge that was queried and did not exist. Therefore, all cycles that appear in the augmenting structures in Q_r consist of edges that either have never been queried, or have been queried and exist. Therefore, each augmenting structure exists with probability at least $p^{k \cdot s_{\eta,k}}$. So, conditioned on $Q_1, \ldots, Q_{r-1}$, the expected increase in the size of the solution at Step 6 is:

$$\begin{aligned}
\mathop{\mathbb{E}}_{Q_r}[|B_r|] - |B_{r-1}| \geq p^{k \cdot s_{\eta,k}}|Q_r| &\geq \frac{p^{k \cdot s_{\eta,k}}}{k \cdot s_{\eta,k}} \left(|K(A_r)| - \frac{|B_{r-1}|}{\frac{2}{k} - \eta} \right) \\
&\geq \gamma \left(\left(\frac{2}{k} - \eta \right) |K(A_r)| - |B_{r-1}| \right) \\
&\geq \gamma \left(\left(\frac{2}{k} - \eta \right) \overline{K}(A \mid Q_1, \ldots, Q_{r-1}) - |B_{r-1}| \right),
\end{aligned}$$

where the last inequality follows by the fact that $|K(A_r)| \geq \overline{K}(A \mid Q_1, \ldots, Q_{r-1})$. Rearranging the above proves the claim. □

We are now ready to prove Theorem 11.7.1.

Proof of Theorem 11.7.1. Let us start with a technical observation, that for every r,

$$\mathop{\mathbb{E}}_{Q_{r-1}}[\overline{K}(A \mid Q_1, \ldots, Q_{r-1})] = \overline{K}(A \mid Q_1, \ldots, Q_{r-2}). \tag{11.5}$$

Using Lemma 11.7.3 on the Rth step of Algorithm 11.4, and conditioning on $Q_1, \ldots, Q_{R-1}$ we have that

$$\mathop{\mathbb{E}}_{Q_R}[|B_R|] \geq (1-\gamma)|B_{R-1}| + \gamma \left(\frac{2}{k} - \eta \right) \overline{K}(A \mid Q_1, \ldots, Q_{R-1}).$$

Taking expectation over Q_{R-1} in the above inequality, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{Q_{R-1},Q_R}[|B_R|] &\geq (1-\gamma)\mathop{\mathbb{E}}_{Q_{R-1}}[|B_{R-1}|] + \gamma\left(\frac{2}{k}-\eta\right)\mathop{\mathbb{E}}_{Q_{R-1}}\left[\overline{K}(A \mid Q_1,\ldots,Q_{R-1})\right] \\
&\geq (1-\gamma)\mathop{\mathbb{E}}_{Q_{R-1}}[|B_{R-1}|] + \gamma\left(\frac{2}{k}-\eta\right)\overline{K}(A \mid Q_1,\ldots,Q_{R-2}) \\
&\geq (1-\gamma)\left((1-\gamma)|B_{R-2}| + \gamma\left(\frac{2}{k}-\eta\right)\overline{K}(A|Q_1,\ldots,Q_{R-2})\right) \\
&\quad + \gamma\left(\frac{2}{k}-\eta\right)\overline{K}(A \mid Q_1,\ldots,Q_{R-2}) \\
&\geq (1-\gamma)^2|B_{R-2}| + \gamma\left(\frac{2}{k}-\eta\right)(1+(1-\gamma))\overline{K}(A|Q_1,\ldots,Q_{R-2}),
\end{aligned}$$

where the second transition is by Equation 11.5, the third transition is due to applying Lemma 11.7.3 on the $(R-1)$th step. Repeating the steps above, by sequentially taking expectation over Q_{R-2} through Q_1, and applying Lemma 11.7.3 and Equation 11.5 at each step, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{Q_1,\ldots,Q_R}[|B_R|] &\geq (1-\gamma)^R|B_0| + \gamma\left(\frac{2}{k}-\eta\right)(1+(1-\gamma)+\ldots,(1-\gamma)^{R-1})\overline{K}(A) \\
&\geq \gamma\left(\frac{2}{k}-\eta\right)(1-(1-\gamma)^R)\overline{K}(A).
\end{aligned}$$

Note that, when $\eta = \frac{\epsilon}{k}$ and $R = \frac{1}{\gamma}\log(\frac{2}{\epsilon}) = \frac{\left(\frac{2}{k}-\eta\right)k\cdot s_{\eta,k}}{p^{k\cdot s_{\eta,k}}}\log(\frac{2}{\epsilon})$, we have

$$\gamma\left(\frac{2}{k}-\eta\right)(1-(1-\gamma)^R) \geq \frac{2}{k}\left(1-\frac{\eta k}{2}\right)(1-(1-\gamma)^R) \geq \frac{2}{k}\left(1-\frac{\epsilon}{2}\right)\left(1-\frac{\epsilon}{2}\right) \geq \frac{2}{k}(1-\epsilon).$$

Therefore, $\mathbb{E}_{Q_1,\ldots,Q_R}[|B_R|] \geq \frac{2}{k}(1-\epsilon)\overline{K}(A)$. We complete the proof by noting that since every cycle in A (and by extension B_R) has between 2 to k vertices, the resulting approximation ratio for the optimal number of vertices covered is

$$\mathop{\mathbb{E}}_{Q_1,\ldots,Q_R}\left[\sum_{c\in B_R}|c|\right] \geq \frac{4}{k^2}(1-\epsilon)\overline{V}(A).$$

□

11.8 Experimental Results

Our theoretical results show that our adaptive and non-adaptive algorithms recover $(1-\epsilon)$ and $(\frac{1}{2}-\epsilon)$ fraction of the omniscient optimum matching using $R = O_{\epsilon,p}(1)$ queries per vertex, respectively. While R is a constant regardless of the number of vertices, its dependence on ϵ and p may lead to values of R that are impractical for a kidney exchange platform. To bridge this

gap, we use empirical simulations from two kidney exchange compatibility graph distributions to show that our algorithms perform well in practice even for a number of per-vertex queries that is as low as $R \leq 5$.

The first distribution, due to Saidman et al. [242], was designed to mimic the characteristics of a nationwide exchange in the United States in steady state. Fielded kidney exchanges have not yet reached that point, though; with this in mind, we also include results on *real* kidney exchange compatibility graphs drawn from the first 169 match runs of the UNOS nationwide kidney exchange. While these two families of graphs differ substantially, we find that even a small number R of non-adaptive rounds, followed by a single period during which only those edges selected during the R rounds are queried, results in large gains relative to the omniscient matching.

As is common in the kidney exchange literature, in the rest of this section we will loosely use the term "matching" to refer to both 2-cycle packing (equivalent to the traditional definition of matching, where two vertices connected by directed edges are translated to two vertices connected by a single undirected edge) and k-cycle packing, possibly with the inclusion of altruist-initiated chains.

This section does not directly test the algorithms presented in this chapter. For the 2-cycles-only case, we do directly implement Algorithm 11.2. However, for the cases involving longer cycles and/or chains, we do not restrict ourselves to polynomial-time algorithms (unlike in the theoretical part of this chapter), instead choosing to optimally solve matching problems using integer programming during each round, as well as for the final matching and for the omniscient benchmark matching. This decision is informed by the current practice in kidney exchange, where computational resources are much less of a problem than human or monetary resources—of which the latter two are necessary for querying edges.

In our experiments, the planning of which edges to query proceeds in rounds as follows. Each round of matching calls as a subsolver the matching algorithm presented by Dickerson et al. [107], which includes edge failure probabilities in the optimization objective to provide a maximum-discounted-utility matching. The set of cycles and chains present in a round's discounted matching are added to a set of edges to query, and then those cycles and chains are constrained from appearing in future rounds. After all rounds are completed, this set of edges is queried, and a final maximum discounted utility matching is compared against an omniscient matching that knows the set of non-failing edges up front.

11.8.1 Experiments on dense generated graphs

We begin by looking at graphs drawn from a distribution due to Saidman et al. [242], hereafter referred to as "the Saidman generator." This generator takes into account the blood types of patients and donors (such that the distribution is drawn from the general United States population), as well as three levels of PRA and various other medical characteristics of patients and donors that may affect the existence of an edge. Fielded kidney exchanges currently do not uniformly sample their pairs from the set of all needy patients and able donors in the US, as assumed by the Saidman generator; rather, exchanges tend to get hard-to-match patients who have not received an organ through other means. Because of this, the Saidman generator tends to produce compatibility graphs that are significantly denser than those seen in fielded kidney exchanges

today (see, e.g., [16, 17]).

Figure 11.4 presents the fraction of the omniscient objective achieved by $R \in \{0, 1, \ldots, 5\}$ non-adaptive rounds of edge testing for generated graphs with 250 patient-donor pairs and no altruistic donors, constrained to 2-cycles only (left) and both 2- and 3-cycles (right). Note that the case $R = 0$ corresponds to no edge testing, where a maximum discounted utility matching is determined by the optimization method of Dickerson et al. [107] and then compared directly to the omniscient matching. The x-axis varies the uniform edge failure rate f from 0.0, where edges do not fail, to 0.9, where edges only succeed with a 10% probability. Given an edge failure rate of f in the figures below, we can translate to the p used in the theoretical section of the chapter as follows: a 2-cycle in a matching represents both directions of an edge and therefore exists with probability $p_{\text{2-cycle}} = (1 - f)^2$, while an edge in a 3-cycle packing only represents a single direction of compatibility and exists with probability $p_{\text{3-cycle}} = 1 - f$ while a 3-cycle exists with probability $(1 - f)^3$.

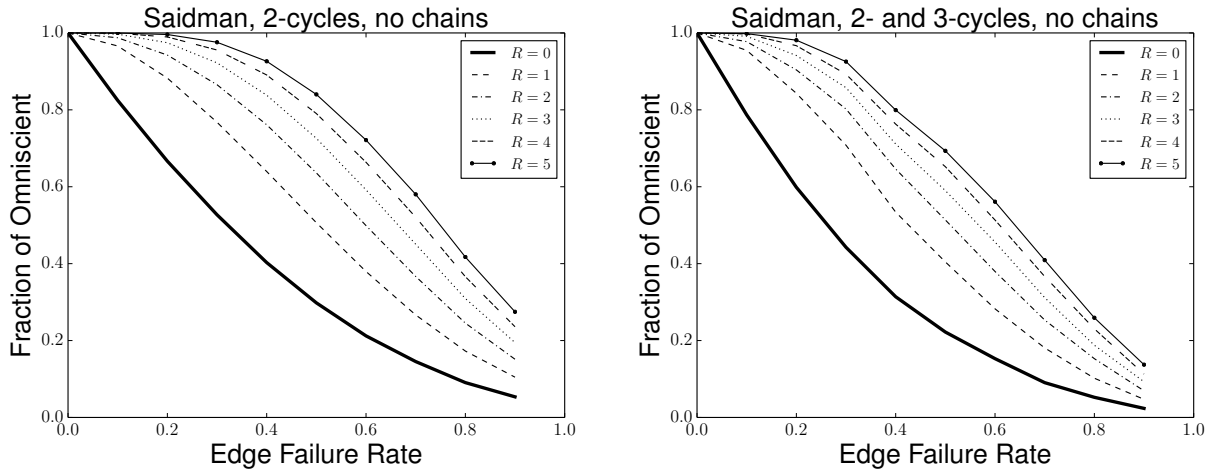


Figure 11.4: Saidman generator graphs constrained to 2-cycles only (left) and both 2- and 3-cycles (right).

The utility of even a small number of edge queries is evident in Figure 11.4. Just a single round of testing ($R = 1$) results in 50.6% of omniscient—compared to just 29.8% with no edge testing—for edge failure probability $f = 0.5$ in the 2-cycle case, and there are similar gains in the 2- and 3-cycle case. For the same failure rate, setting $R = 5$ captures 84.0% of the omnsicient 2-cycle matching and 69.3% in the 2- and 3-cycle case—compared to just 22.2% when no edges are queried. Interestingly, we found no statistical difference between non-adaptive and adaptive matching on these graphs.

11.8.2 Experiments on real match runs from the UNOS nationwide kidney exchange

We now analyze the effect of querying a small number of edges per vertex on graphs drawn from the real world. Specifically, we use the first 169 match runs of the UNOS nationwide kidney exchange, which began matching in October 2010 on a monthly basis and now includes 153

transplant centers—that is, 66% of the centers in the U.S.—and performs match runs twice per week. These graphs, as with other fielded kidney exchanges [17], are substantially less dense than those produced by the Saidman generator. This disparity between generated and real graphs has led to different theoretical results (e.g., efficient matching does not require long chains in a deterministic dense model [14, 105] but does in a sparse model [16, 109]) and empirical results (both in terms of match composition and experimental tractability [9, 86, 129]) in the past—a trend that continues here.

Figure 11.5 shows the fraction of the omniscient 2-cycle and 2-cycle with chains match size achieved by using only 2-cycles or both 2-cycles and chains and some small number of non-adaptive edge query rounds $R \in \{0, 1, \ldots, 5\}$. For each of the 169 pre-test compatibility graphs and each of edge failure rates, 50 different ground truth compatibility graphs were generated. Chains can partially execute; that is, if the third edge in a chain of length 3 fails, then we include all successful edges (in this case, 2 edges) until that point in the final matching. More of the omniscient matching is achieved (even for the $R = 0$ case) on these real-world graphs than on those from the Saidman generator presented in Section 11.8.1. Still, the gain realized even by a small number of edge query rounds is stark, with $R = 5$ achieving over 90% of the omniscient objective for every failure rate in the 2-cycles-only case, and over 75% of the omniscient objective when chains are included (and typically much more).

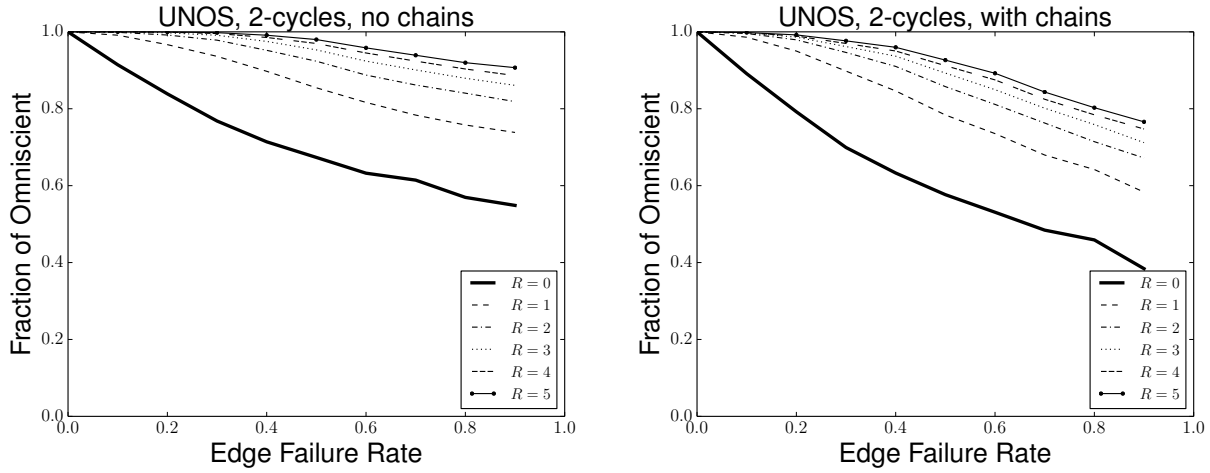


Figure 11.5: Real UNOS match runs constrained to 2-cycles (left) and both 2-cycles and chains (right).

Figure 11.6 expands these results to the case with 2- and 3-cycles, both without and with chains. Slightly less of the omniscient matching objective is achieved across the board, but the overall increases due to $R \in \{1, \ldots, 5\}$ non-adaptive rounds of testing is once again prominent. Interestingly, we did not see a significant difference in results for adaptive and non-adaptive edge testing on the UNOS family of graphs, either.

Next we consider the above experiments again, only this time including in the analysis empty omniscient matchings. If an omniscient matching is empty, then our algorithm will achieve at most zero matches as well. Previously, we removed these cases from the experimental analysis because achieving zero matches (using any method) out of zero possible matches trivially

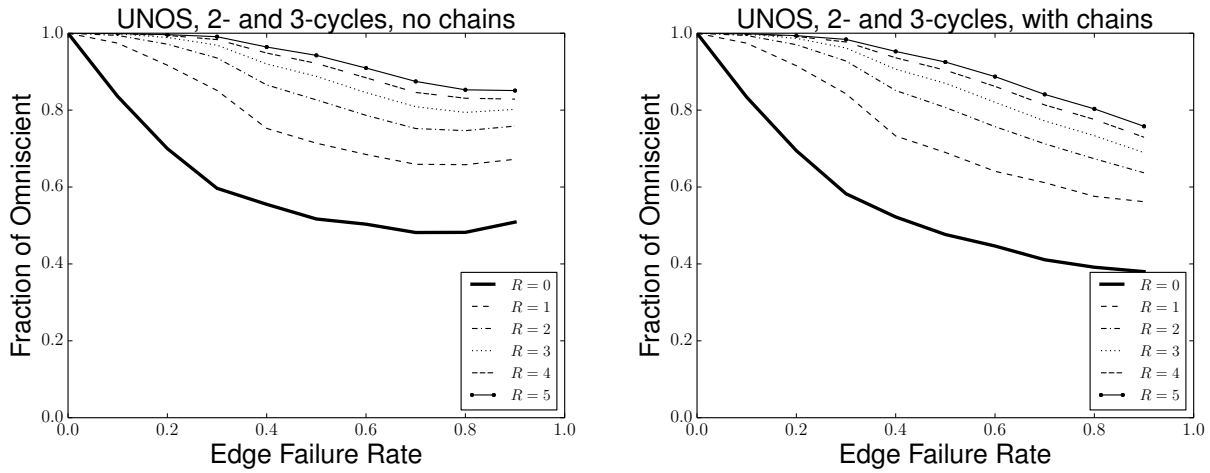


Figure 11.6: Real UNOS match runs with 2- and 3-cycles and no chains (left) and with chains (right).

achieves 100% of the omniscient matching; by not including those cases, we provided a more conservative experimental analysis. Here, we include those cases and rerun the analysis.

Figure 11.7 mimics Figure 11.5. It shows results for 2-cycle matching on the UNOS compatibility graphs, without chains (left) and with chains (right), for $R \in \{0, 1, \ldots, 5\}$ and varying levels of $f \in \{0, 0.1, \ldots, 0.9\}$. We witness a marked increase in the fraction of omniscient matching achieved as f gets close to 0.9; this is due to the relatively sparse UNOS graphs admitting no matchings for high failure rates.

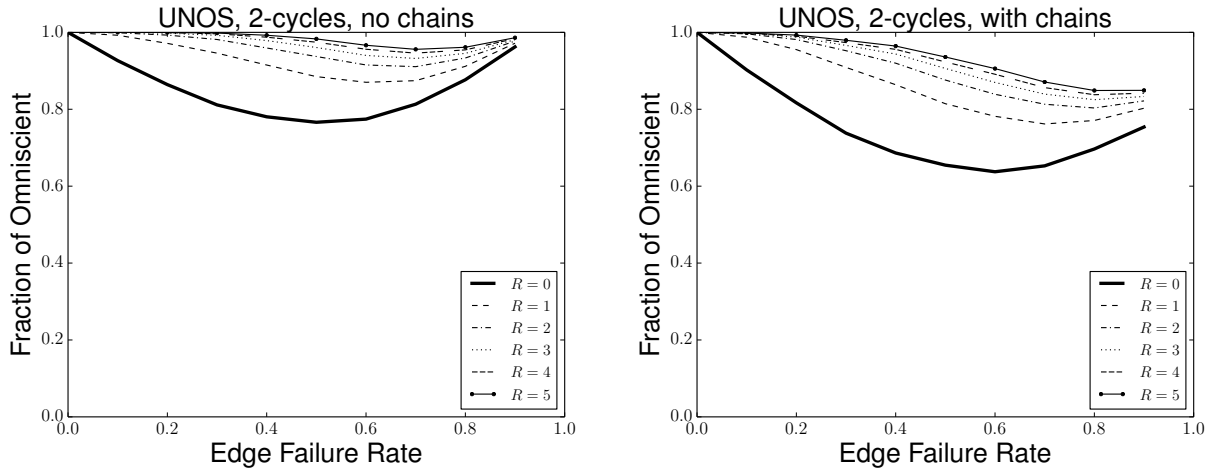


Figure 11.7: Real UNOS match runs, restricted matching of 2-cycles only, without chains (left) and with chains (right), including zero-sized omnsicient matchings.

Figure 11.8 shows the same experiments as Figure 11.7, only this time allowing both 2- and 3-cycles, without (left) and with (right) chains. It corresponds to Figure 11.6 and exhibits similar but weaker behavior to Figure 11.7 for high failure rates. This demonstrates the power of

including 3-cycles in the matching algorithm—we see that far fewer compatibility graphs admit no matchings under this less-restrictive matching policy.

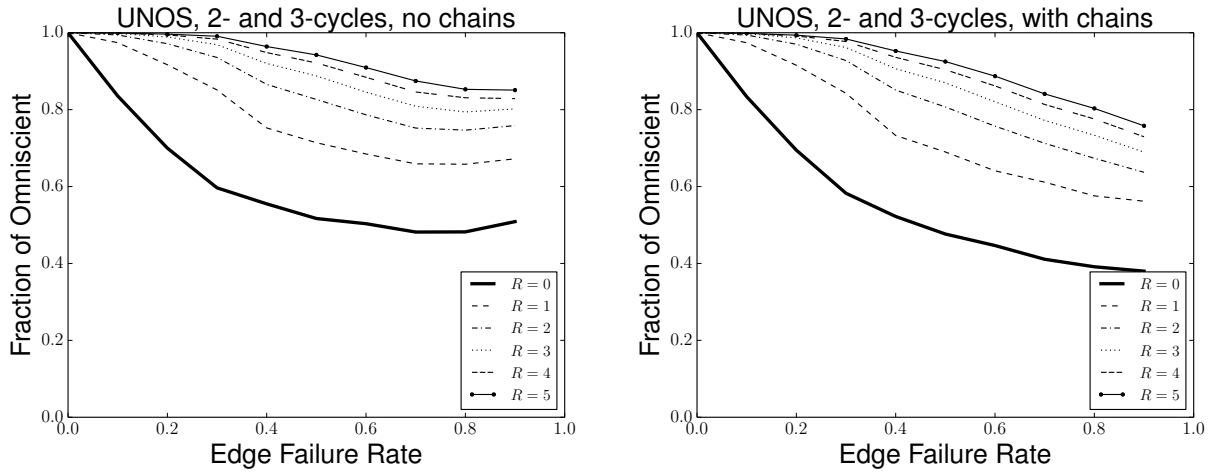


Figure 11.8: Real UNOS match runs, matching with 2- and 3-cycles, without chains (left) and with chains (right), including zero-sized omnsicient matchings.

11.9 Discussion & future research

In this chapter, we addressed stochastic matching and its generalization to k-cycle packing from both a theoretical and experimental point of view. For the stochastic matching problem, we designed an *adaptive* algorithm that queries only a constant number of edges per vertex and achieves a $(1-\epsilon)$ fraction of the omniscient solution, for an arbitrarily small $\epsilon > 0$—and performs the queries in only a constant number of rounds. We complemented this result with a *non-adaptive* algorithm that achieves a $(0.5-\epsilon)$ fraction of the omniscient optimum.

We then extended our results to the more general problem of *stochastic k-cycle packing* by designing an adaptive algorithm that achieves a $(\frac{2}{k}-\epsilon)$ fraction (res. $(\frac{4}{k^2}-\epsilon)$ fraction) of the cardinality of (res. number of vertices covered in) the omniscient optimal solution, again with only $O(1)$ queries per element. This guarantee is quite close to the best known polynomial-time approximation ratio of $\frac{3}{k+1}-\epsilon$ for the cardinality of the optimal k-cycle packing in the standard *non-stochastic* setting [126].

We adapted these algorithms to the kidney exchange problem and, on both generated and real data from the first 169 runs of the UNOS US nationwide kidney exchange, explored the effect of a small number of edge query rounds on matching performance. In both cases—but especially on the real data—a very small number of non-adaptive edge queries per donor-patient pair results in large gains in expected successful matches across a wide range of edge failure probabilities.

In the theoretical part of this chapter, we considered a setting where every edge e exists with probability $p_e \geq p$ for a constant value of p and gave algorithms that queried a number of edges that increased as $p \to 0$. In kidney exchange, however, a small fraction of patients may be highly sensitized; there is a low probability that their crossmatch test with potentially compatible donors

would be successful. This gives rise to compatibility graphs that include a small fraction of edges with success probability $p_e \approx 0$. In this case, setting $p = \min_{e \in E} p_e$ in our theoretical results would require a large number of tests per vertex, leading to an impractical algorithm that tests every edge and finds the omniscient optimal solution. To avoid this problem, we note that when the number of edges with small p_e is small, ignoring them affects the size of the omniscient optimal solution only to a small degree. This allows us to essentially recover our theoretical approximation guarantees. Of course, in practice these edges are not ignored, rather they may receive specialized treatments, hence allowing for an even better outcome.

11.9.1 Open theoretical problems

Three main open theoretical problems remain open. First, our *adaptive* algorithm for the matching setting achieves a $(1-\epsilon)$-approximation in $O(1)$ rounds and using $O(1)$ queries per vertex. Is there a *non-adaptive algorithm* that achieves the same guarantee? Such an algorithm would make the practical message of the theoretical results even more appealing: instead of changing the *status quo* in two ways—more rounds of crossmatch tests, more tests per patient—we would only need to change it in the latter way.

Second, for the case of optimal cardinality k-cycle packing, we achieve a $(\frac{2}{k}-\epsilon)$-approximation using $O(n)$ queries—in polynomial time. In kidney exchange, however, our scarcest resource is the ability to query edges. In particular, computational hardness is circumvented in many cases through integer programming techniques [3, 86, 108]. Therefore, it would be interesting to see if there is an exponential-time adaptive algorithm for k-cycle packing that requires $O(1)$ rounds and $O(n)$ queries, and achieves $(1-\epsilon)$-approximation to the omniscient optimum. A positive answer would require a new approach, because ours is inherently constrained to constant-size augmenting structures, which cannot yield an approximation ratio better than $\frac{2}{k} - \epsilon$, even if we could compute optimal solutions to k-cycle packing [161].

Third, while we provided an adaptive algorithm for stochastic k-cycle packing for the general case of $k \geq 2$, our non-adaptive results were restricted to the case of $k = 2$. A natural question is whether there exists a non-adaptive algorithm with a good approximation guarantee for stochastic k-cycle packing when $k > 2$. One of the key ingredients in the analysis of our non-adaptive stochastic matching algorithm was to show that the benefit we drew from a new matching (whose edges were to be tested at the end of the algorithm) was (1) a large fraction of the size of the matching in the remaining graph and (2) independent of the outcome of the queries in the earlier matchings (See Equation 11.4). For the case of $k = 2$, these properties hold because the optimal matching in the remaining graph at step r is a large fraction of the total matching and the matching at step r shares no edges with prior matchings. For the case of $k > 2$, however, to assure that property (2) holds, we need to choose a k-cycle packing at step r using only those cycles that share no edges with the k-cycle packings in steps $1, \ldots, r-1$. Therefore, at every step we need to remove from consideration all cycles that share an edge with an earlier packing. However, doing so results in a graph that has a small (or no) cycle packing, invalidating property (1). We conclude that new algorithms and techniques may be needed for the non-adaptive stochastic k-cycle packing problem.

11.9.2 Discussion of policy implications of experimental results

Policy decisions in kidney exchange have been linked to economic and computational studies since before the first large-scale exchange was fielded in 2003–2004 [232, 233]. A feedback loop exists between the reality of fielded exchanges—now not only in the United States but internationally as well—and the theoretical and empirical models that inform their operation, such that the latter has grown substantially closer to accurately representing the former in recent years. That said, many gaps still exist between the mathematical models used in kidney exchange studies and the systems that actually provide matches on a day-to-day basis.

Better approaches are often not adopted quickly, if at all, by exchanges. One reason for this is complexity—and not in the computational sense. Humans—doctors, lawyers, and other policymakers who are not necessarily versed in optimization, economics, or computer science—and the organizations they represent understandably wish to understand the workings of an exchange's matching policy. The techniques described in this chapter are particularly exciting in that they are quite easy to explain in accessible language and they involve only mild changes to the status quo. At a high level, we are proposing to test some small number of promising potential matches for some subset of patient-donor pairs in a pool. As Section 11.8.2 shows, even a *single* extra edge test per pair will produce substantially better results.

Any new policy for kidney exchange has to address three practical restrictions in this space: (i) the monetary cost of crossmatches, (ii) the number of crossmatches that can be performed per person, as there is an inherent limit on the amount of blood that can be drawn from a person, and (iii) the time it takes to find the matches, as time plays a major role in the health of patients and crossmatches become less accurate as time passes and the results become stale. For both our non-adaptive and adaptive algorithms, even a very small number of rounds ($R \leq 5$) results in a very large gain in the objective. This is easily within the limits of considerations (i) and (ii) above. Our non-adaptive algorithm performs all chosen crossmatches in parallel, so the time taken by this method is similar to the current approach. Our adaptive algorithm, in practice, can be implemented by a one-time retrieval of R rounds worth of blood from each donor-patient pair, then sending that blood to a central laboratory. Most crossmatches are performed via an "immediate spin", where the bloods are mixed together and either coagulate (which is bad) or do not (which is good). These tests are very fast, so a small number of rounds could be performed in a single day (assuming that tests in the same round are performed in parallel). Therefore, the timing constraint (iii) is not an issue for small R (such as that used in our experiments) for the adaptive algorithm.

More extensive studies would need to be undertaken before an exact policy recommendation can be made. These studies could take factors like the monetary cost of an extra crossmatch test or variability in testing prowess across different medical laboratories into account explicitly during the optimization process. Various prioritization schemes could also be implemented to help, for example, hard-to-match pairs find a feasible match by assigning them a higher edge query budget than easier-to-match pairs. Moreover, there is a need for a closer look at other uncertainties in kidney exchange, such as the dynamic nature of participation of donors and patients, and how they interact with our proposed algorithms. But, the positive theoretical results presented in this chapter, combined with the promising experimental results on real data, provide a firm basis and motivation for this type of policy analysis in the future.

Chapter 12

Individually Rational Multi-Hospital Kidney Exchange

12.1 Introduction

As kidney exchange programs have become more prevalent, multi-hospital are being deployed at the national and international levels. These programs create an environment where patient-donor pairs can be matched across different hospitals and exchanges. The reason why such programs are desirable is clear: By increasing the number of potential matches that a donor-patient pair can be matched to—from those in a single hospital to all hospitals—we can match more patients to compatible donors. Despite this, there are serious concerns regarding the practicality of such systems. Based on their work with practicioners, Ashlagi and Roth [14] convincingly argue that as kidney exchange programs outgrow their regional origins, the incentives of hospitals and local exchanges—which have little to no interaction outside of the kidney exchange program—become misaligned. In particular, hospitals cannot be certain that if they opt into a multi-hospital kidney exchange program, which optimizes overall efficiency, their patients would be better off overall than under the optimal *internal* matching (which relies only on donor-patient pairs associated with the hospital). This is called lack of *individual rationality*.

In this chapter, we study individual rationality of matching markets, including multi-hospital kidney exchange programs, more closely. Specifically, we study situations where the vertices of the graph are partitioned between a set of players, and each player is interested in matching as many of *his own* vertices as possible. An *individually rational matching* is one that matches at least as many vertices of each player as he can match on his own.

How bad is lack of individual rationality in kidney exchange? A bad example (due to Ashlagi

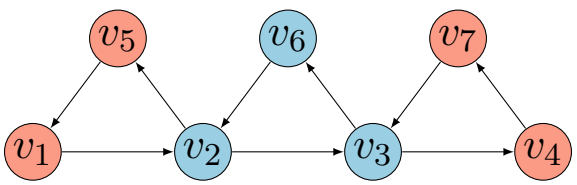


Figure 12.1: A compatibility graph where individual rationality fails.

and Roth) is given in Figure 12.1: the maximum cardinality matching selects the 3-cycles $\{v_1, v_2, v_5\}$ and $\{v_3, v_4, v_7\}$, but the blue player can do better by internally matching the single 3-cycle $\{v_2, v_3, v_6\}$. That is, the global matching matches twice as many nodes than the number of nodes collectively covered by the local matchings. Furthermore, the unique individually rational matching in this graph is the internal matching. So, ensuring that the matching is individually rational comes at the cost of losing half of the size of the globally optimal matching.

12.1.1 Our Approach

In our work, we take a new perspective on individual rationality. Our goal is to analytically demonstrate that, in fact, individual rationality (or an almost perfect approximation thereof) can be achieved with nearly no loss of efficiency under mild conditions. Our key insight is that it suffices to assume that each vertex of the graph is owned by a *random player*.

In more detail, we consider an arbitrary graph with n vertices $v_1, \ldots, v_n$, and a set of k players $1, \ldots, k$. For each vertex, we draw its owner independently from the probability distribution $p_1, \ldots, p_k$ over the players, that is, each vertex is assigned to player i with probability p_i. In the kidney exchange setting, for example, the rationale is very simple: the graph represents medical compatibility information, and there is no special reason why a patient-donor pair with particular medical characteristics would belong to a particular hospital — the probability of that happening depends chiefly on the size of the hospital.

To see how randomization helps, let us revisit the example given in Figure 12.1, and suppose that the two players (red and blue) have probability $1/2$ each: $p_1 = p_2 = 1/2$. The expected utility of a player under the maximum cardinality matching is $6/2 = 3$. In contrast, a straightforward upper bound on the expected cardinality of an internal matching can be derived by observing that each of the three 3-cycles is owned by a single player with probability $1/8$ and adds at most 3 to the cardinality of the matching, leading to an upper bound of $3 \cdot 3 \cdot (1/8) = 9/8$. Now, suppose we made t copies of the graph of Figure 12.1, for a large t; then a simple measure concentration argument would imply that it is very likely that each player is better off in the optimal solution than he is working alone.

Our goal is to establish this phenomenon in some generality. Indeed, our qualitative message (a few technical caveats apply) is that

> ... *in an* arbitrary *graph, under a random assignment of vertices to players, it is likely that* any *fixed optimal matching is individually rational, up to lower order terms, for each player; and there is a* practical *mechanism that yields an individually rational matching that is likely to be optimal up to lower order terms.*

12.1.2 Our Results and Techniques

In Section 12.2, we formalize the first part of the above statement. Specifically, we prove the following theorem:

Theorem 12.2.2 (informally stated). *Let G be a directed graph, and let $\textsc{Opt}(G)$ be the set of vertices matched under a specific maximum cardinality matching on G. Assume that one of the following conditions holds:*

1. *Matchings are restricted to 2-cycles, and $p_i \leq 1/2$ for each player i, or*

2. *Matchings are restricted to cycles of constant length, and for each player i, $1/p_i$ is an integer.*

Then for each player $i \in [k]$, the difference between the size of his optimal internal matching, and his share of $\textsc{Opt}(G)$*, is at most $O(\sqrt{|\textsc{Opt}(G)| \cdot \ln(k/\delta)})$ with probability $1 - \delta$.*

The theorem's first case deals with 2-cycles, a common abstraction for kidney exchange in theoretical studies [6, 18, 19, 55, 58, 68, 233, 257]. Of course in this case there always *exists* an optimal and individually rational matching (find the optimal internal matchings and then add augmenting paths), but nonetheless this statement is appealing because it applies to *any* optimal solution that the exchange — which might also be optimizing some secondary objective — might produce. Also note that this case is essentially unrestrictive in terms of the probability distribution. The second case is the opposite: its assumption of constant length cycles is essentially unrestrictive, as chains can be represented as cycles by adding an edge from every patient-donor pair to every altruistic donor; and major kidney exchanges — such as the US national program, run by the United Network for Organ Sharing (UNOS) — use only cycles of length at most 3, and chains of length at most 4 [14, 16, 105, 234]. But the assumption regarding the probability distribution is, of course, somewhat restrictive. Note, however, that probabilities can be "rounded" at a cost, as we discuss later; and that the natural case of equal probabilities is captured by the second case.

The proof of Theorem 12.2.2 relies on two main ingredients. The first is the claim that the expected size of the maximum internal matching of player i is at most a p_i fraction of the optimal (global) matching. This statement is almost trivial in Case 2; to establish it in Case 1, we decompose the maximum cardinality matching via the Edmonds-Gallai Decomposition [115], and show that the inequality holds for each component separately.

The second ingredient is the concentration of the cardinality of the optimal internal matching of each player around its expectation. To this end, we leverage machinery from modern probability theory that is little known in theoretical computer science, including a concentration inequality for so-called *self-bounding functions* [64].

The power of Theorem 12.2.2 is that it applies to *any* maximum cardinality matching. In the context of kidney exchange, the theorem captures the matching algorithms currently in use (including the ones employed by UNOS); its conceptual message is that hospitals need not worry about opting into kidney exchange programs, even under the *status quo*.

By contrast, in Section 12.3 we give the designer more power in choosing the matching, with the goal of constructing a mechanism that is (perfectly) individually rational, and almost optimal. As noted earlier, this is quite trivial when only 2-cycles are allowed, as there always exists an optimal, individually rational matching. When longer cycles are allowed, we can derive the following corollary from the proof of Theorem 12.2.2.

Corollary 12.3.1 (informally stated). *Let G be a directed graph with n vertices, and let* $\textsc{Opt}(G)$ *be the set of vertices matched under a specific maximum cardinality matching on G. Suppose that matchings are restricted to cycles of constant length, and $p_i = p_j$ for any two players i, j. Then, with probability $1 - \delta$, there exists a matching that is individually rational for each player i, and has maximum cardinality up to $O(k\sqrt{|\textsc{Opt}(G)| \cdot \ln(k/\delta)})$.*

Importantly, such an individually rational and almost optimal matching can be found with a *practical*[1] mechanism: (i) compute a maximum cardinality matching, (ii) any player who wishes to work alone is allowed to defect.

Furthermore, we show that our results are tight. Among other things, we construct an example with two players and cycles up to length 3 such that, with constant probability, *any* individually rational matching is smaller than the optimal matching by $\Omega(\sqrt{|\textsc{Opt}(G)|})$.

12.1.3 Related Work

The two papers that are most closely related to ours are the ones by Ashlagi and Roth [14] and Toulis and Parkes [257]. Ashlagi and Roth show that under some technical assumptions, and under a random graph model of kidney exchange, large random graphs admit an individually rational matching that is optimal up to a certain constant fraction of the number of vertices, with high probability. Toulis and Parkes [257] independently study a very similar random graph model (it does make different assumptions about the size of hospitals), and obtain a similar result regarding individual rationality.

These important results have inspired our own work, but — in addition to a number of significant technical advantages[2] — we believe our high-level approach is significantly more compelling. In a nutshell, the random graph model studied by Ashlagi and Roth [14] and Toulis and Parkes [257] draws blood types for each donor and patient from a distribution that gives each of the four blood types (O, A, B, and AB) constant probability. For each pair of blood type compatible vertices (e.g., an O donor is blood type compatible with an A patient, but a B donor is not), a directed edge exists with *constant* probability. This model clearly gives rise to very dense graphs; the key to the abovementioned results is that, with high probability, there exist matchings between blood type compatible groups (such as A patient and B donor, and B patient and A donor) that are perfect in the sense that they match all the vertices in the smaller group. Consequently, the structure of the optimal matching can be accurately predicted with high probability. This model has subsequently been employed in several other papers [55, 105].

However, more recent work by Ashlagi and Roth themselves — together with collaborators [16] — introduces a completely different random graph model of kidney exchange, which gives rise to sparse graphs, and better captures some real-world phenomena. This model was later employed by Dickerson et al. [107]. At this point it is fair to say that, on the question of whether random graph models are a valid approach for the analysis of kidney exchange, the jury is still out. But we are convinced that an analysis that holds for *arbitrary* graphs — when it is feasible, as in this chapter — is the right approach.

Individual rationality limits players to two possible strategies: work alone or participate fully. More generally, players can choose to reveal a subset of their vertices, and internally match the rest. Several papers seek to design mechanisms that incentivize players to reveal all their

[1] By "practical" we mean that it can be easily implemented in practice. It is not a polynomial-time algorithm, as computing a maximum cardinality matching is $\mathcal{NP}$-hard when 3-cycles are allowed [3]; but the problem is routinely solved via integer programming.

[2] In contrast to their work, we obtain optimality up to lower-order terms (instead of up to a constant fraction of n), and our results have a good dependence on the number of players (instead of assuming a very large [14] or a very small [257] number) and on the size of the graph (instead of assuming that n goes to infinity).

vertices, either as a dominant strategy [18, 145] or in equilibrium [14, 257]. These known results are quite limited; obtaining stronger results is a central open problem. Our own approach does not seem to extend beyond individual rationality.

Needless to say, matching is a major research topic in theoretical computer science. In particular, there are many papers that are directly motivated by market design applications, especially kidney exchange [4, 42, 77, 87, 130, 136]. These papers are largely orthogonal to our work.

12.2 Optimal Matchings Are Almost Individually Rational

Designing and implementing new matching mechanisms can require significant changes to current policies and deployed algorithms. In this section, we show that even without any changes to the existing (optimal) matching mechanisms — at least in the case of kidney exchange — it is likely that each player matches almost as many vertices as what he could have obtained on his own.

Consider an arbitrary directed graph G with n vertices. Recall that for each player $i \in [k]$ with corresponding probability p_i, the player owns each vertex with probability p_i, independently. We denote by $H_i \sim_{p_i} G$ the random subgraph of player i, which is a subgraph of G induced by assigning each vertex to i with probability p_i. We suppress p_i from this notation when it is clear from the context. We use $\textsc{Opt}(G)$ to denote the set of *vertices* of an arbitrary but fixed matching of G. Furthermore, $\textsc{Opt}(G) \restriction H_i$ denotes the restriction of $\textsc{Opt}(G)$ to subgraph H_i, i.e., the *vertices* of H_i that are matched under $\textsc{Opt}(G)$. Therefore, to compare the size of the internal matching of H_i with the number of vertices of H_i that are matched under the global matching, we compare $|\textsc{Opt}(H_i)|$ to $|\textsc{Opt}(G) \restriction H_i|$, and show that these values are within $\tilde{O}(\sqrt{|\textsc{Opt}(G)|})$ of one another.

Let us first describe a graph in which, with a constant probability, a player's internal matching is larger by $\Omega(\sqrt{|\textsc{Opt}(G)|})$ than the player's share of any fixed optimal matching.

Example 12.2.1. *Suppose one of the players has probability $p = \frac{1}{2}$, and consider a graph that consists of $n/\log(n)$ stars, each with $\log(n)$ vertices that are connected to the center via 2-cycles. Fix an optimal global matching, $\textsc{Opt}(G)$, and note that $|\textsc{Opt}(G)| = 2n/\log(n)$. We informally argue that there is a constant $c > 0$ such that*

$$\Pr_{H \sim_p G}\left[|\textsc{Opt}(H)| - |\textsc{Opt}(G) \restriction H| > \Omega(\sqrt{|\textsc{Opt}(G)|})\right] > c.$$

Indeed, let us consider the subgraph internal to the player, $H \sim_p G$. While the expected number of centers in H is $\frac{1}{4}|\textsc{Opt}(G)|$, it is easy to see (by looking up the standard deviation of the binomial distribution) that, with constant probability, H includes $t = \frac{1}{4}|\textsc{Opt}(G)| + \Theta(\sqrt{|\textsc{Opt}(G)|})$ centers. Moreover, with probability $\frac{1}{2}$, H includes no more than half of the non-center vertices matched by $\textsc{Opt}(G)$. If both events occur, $|\textsc{Opt}(G) \restriction H| \leq t + \frac{1}{4}|\textsc{Opt}(G)|$, where t of the matched vertices correspond to the center vertices and at most $\frac{1}{4}|\textsc{Opt}(G)|$ vertices correspond to non-center vertices of H that coincide with $\textsc{Opt}(G)$. On the other hand, each star is large enough so that with constant probability H includes at least one non-center vertex in each

star. In that case, for every center vertex in H, $\textsc{Opt}(H)$ gets two matched vertices. Therefore, $|\textsc{Opt}(H)| \geq 2t$ internally. It follows that the player can gain an additional $\Theta(\sqrt{|\textsc{Opt}(G)|})$ matched vertices when deviating from a fixed global optimal matching.

Our main result shows that Example 12.2.1 is asymptotically tight.

Theorem 12.2.2. *Let G be a directed graph and let $\textsc{Opt}(G)$ be the set of vertices matched under some fixed maximum cardinality matching on G. Assume that one of the following conditions holds:*

1. *Matchings are restricted to 2-cycles, and $p_i \leq 1/2$ for each player $i \in [k]$, or*
2. *Matchings are restricted to cycles of constant length, and for each player $i \in [k]$, $1/p_i$ is an integer.*

Then for any $\delta > 0$,

$$\Pr_{H_i \sim_{p_i} G}\left[\forall i \in [k],\ |\textsc{Opt}(H_i)| - |\textsc{Opt}(G) \restriction H_i| < O\left(\sqrt{|\textsc{Opt}(G)| \ln \frac{k}{\delta}}\right)\right] \geq 1 - \delta.$$

The proof of Theorem 12.2.2 involves two main lemmas. The first shows that in expectation $|\textsc{Opt}(H_i)|$ is at most $|\textsc{Opt}(G) \restriction H_i|$. The second asserts that $|\textsc{Opt}(H_i)|$ is concentrated nicely around its expectation. We formally state these two lemmas without further ado, but defer their proofs to Section 12.2.1 and Section 12.2.2, respectively.

Lemma 12.2.3. *Let G be a directed graph and let $\textsc{Opt}(G)$ be the set of vertices matched under some fixed maximum cardinality matching on G. Then $\mathbb{E}_{H \sim_p G}[|\textsc{Opt}(H)|] \leq p|\textsc{Opt}(G)|$ if (i) matchings are restricted to 2-cycles, and $p \leq 1/2$, or (ii) $1/p$ is an integer.*

Lemma 12.2.4. *Let G be a directed graph and let $\textsc{Opt}(G)$ be the set of vertices matched under some fixed maximum cardinality matching on G. Assume matchings are restricted to cycles of length up to a constant L. Then for any $\delta > 0$, with probability $1 - \delta$ over random choices of $H_i \sim_{p_i} G$, for all $i \in [k]$,*

$$\mathbb{E}_{H_i \sim_{p_i} G}[|\textsc{Opt}(H_i)|] - L\sqrt{2 \cdot \mathbb{E}[|\textsc{Opt}(H_i)|] \ln \frac{2k}{\delta}} < |\textsc{Opt}(H_i)|$$

and

$$\mathbb{E}_{H_i \sim_{p_i} G}[|\textsc{Opt}(H_i)|] + 2L\sqrt{|\textsc{Opt}(G)| \ln \frac{2k}{\delta}} > |\textsc{Opt}(H_i)|.$$

We now easily prove our main result — Theorem 12.2.2 — by directly leveraging the two lemmas we just stated.

Proof of Theorem 12.2.2. Since $\textsc{Opt}(G)$ is fixed and H_i is drawn from G independently of $\textsc{Opt}(G)$, the expected number of vertices player i has in $\textsc{Opt}(G)$ is

$$\mathbb{E}_{H_i \sim_{p_i} G}[|\textsc{Opt}(G) \restriction H_i|] = p_i \cdot |\textsc{Opt}(G)|. \tag{12.1}$$

Moreover, $|\textsc{Opt}(G) \upharpoonright H_i| = \sum_{v \in \textsc{Opt}(G)} \mathbb{I}_{v \in H_i}$, where $\mathbb{I}_{v \in H_i}$ is an indicator variable with value 1 if v is owned by player i and 0 otherwise. So, $\mathbb{I}_{v \in H_i}$ is a random variable that has value 1 with probability p_i, and value 0 otherwise. Using Hoeffding's inequality over $|\textsc{Opt}(G)|$ i.i.d. variables for a fixed $i \in [k]$, as well as Equation (12.1),

$$\Pr_{H_i \sim_{p_i} G}\left[|\textsc{Opt}(G) \upharpoonright H_i| \geq p_i \cdot |\textsc{Opt}(G)| - \sqrt{\frac{1}{2}|\textsc{Opt}(G)| \ln \frac{2k}{\delta}}\right] \geq 1 - \frac{\delta}{2k}. \tag{12.2}$$

Putting this together with Lemmas 12.2.3 and 12.2.4, we have that for all $i \in [k]$, with probability $1 - \delta$,

$$\begin{aligned}
|\textsc{Opt}(H_i)| &\leq \mathop{\mathbb{E}}_{H_i \sim_{p_i} G}[|\textsc{Opt}(H_i)|] + 2L\sqrt{|\textsc{Opt}(G)| \ln \frac{4k}{\delta}} \\
&\leq p_i\,|\textsc{Opt}(G)| + 2L\sqrt{|\textsc{Opt}(G)| \ln \frac{4k}{\delta}} \\
&\leq |\textsc{Opt}(G) \upharpoonright H_i| + (2L+1)\sqrt{|\textsc{Opt}(G)| \ln \frac{4k}{\delta}},
\end{aligned}$$

where the first inequality follows from Lemma 12.2.4 (using $\delta' = \delta/2$), the second from Lemma 12.2.3, and the third from applying Equation (12.2) to each $i \in [k]$. □

Note the logarithmic dependence of Theorem 12.2.2 on the number of players k. A subtle point is that if the number of players is large, some will have a small p_i, which means that the expectation of $|\textsc{Opt}(G) \upharpoonright H_i|$ is small compared to $|\textsc{Opt}(G)|$, by Equation (12.1). In that case, a gain of $\sqrt{|\textsc{Opt}(G)|}$ is significant. Nevertheless, the theorem's conceptual message — that following the global matching is individually rational up to lower order terms — holds for any $p_i = \omega(1/\sqrt{|\textsc{Opt}(G)|})$.

In addition, recall that Theorem 12.2.2 considers two cases, (i) $1/p_i$ is an integer and (ii) $p_i \leq 1/2$ and $\textsc{Opt}(G)$ is restricted to 2-cycles. Importantly, these two assumptions are only needed for Lemma 12.2.3. We conjecture that indeed Lemma 12.2.3 holds for any $p \leq \frac{1}{2}$, whenever $\textsc{Opt}(G)$ is restricted to cycles of constant length — in which case Theorem 12.2.2, too, would hold under this weaker assumption. One might wonder why assuming $p \leq \frac{1}{2}$ is even necessary. But in Appendix 12.4.1 we construct examples that violate the conclusion of Theorem 12.2.2 for certain values of $p > 1/2$.

Finally, we remark that if $1/p_i$ is close to an integer but not itself an integer, one can first round down p_i to the largest $q_i < p_i$ such that $1/q_i$ is an integer, and then apply Theorem 12.2.2. This would give the same result, up to an additional *constant* fraction of $|\textsc{Opt}(G)|$. As p_i becomes smaller the rounding error also diminishes.

12.2.1 Proof of Lemma 12.2.3

First, let us address Case 2 of the lemma. Consider p such that $1/p$ is an integer; $\textsc{Opt}(G)$ may include cycles of any length. Imagine there are $\frac{1}{p}$ players, each with probability p. By symmetry between the players, the expected size of the optimal matching in all subgraphs is

equal. Furthermore, the total number of vertices matched by players individually is at most $\textsc{Opt}(G)$. Therefore,

$$|\textsc{Opt}(G)| \geq \sum_{i=1}^{1/p} \mathop{\mathbb{E}}_{H_i \sim_p G}[|\textsc{Opt}(H_i)|] = \frac{1}{p} \mathop{\mathbb{E}}_{H \sim_p G}[|\textsc{Opt}(H)|]\,,$$

which proves the claim.

In the remainder of this section we focus on Case 1 of Lemma 12.2.3, where the matchings are restricted to 2-cycles and $p \leq 1/2$. For ease of exposition, we treat G as an undirected graph: each directed 2-cycle corresponds to an undirected edge, and we may remove directed edges that are not involved in 2-cycles (as they are useless).

Assume there is a partition $G = G_1 \uplus G_2 \uplus \cdots \uplus G_\ell$ of (the undirected graph) G into edge-disjoint (but not necessarily vertex-disjoint) subgraphs that preserve the size of the optimal matching, i.e.,

$$|\textsc{Opt}(G)| = \sum_{i=1}^{\ell} |\textsc{Opt}(G_i)|. \tag{12.3}$$

Moreover, assume that each of these subgraphs has the property that

$$\mathop{\mathbb{E}}_{H \sim_p G_i}[|\textsc{Opt}(H)|] \leq p\,|\textsc{Opt}(G_i)|. \tag{12.4}$$

Then, the next equation proves that this property also holds for G at the global level. That is,

$$\begin{aligned}
\mathop{\mathbb{E}}_{H \sim_p G}[|\textsc{Opt}(H)|] &\leq \sum_{i=1}^{\ell} \mathop{\mathbb{E}}_{H \sim_p G}[|\textsc{Opt}(H \cap G_i)|] \\
&= \sum_{i=1}^{\ell} \mathop{\mathbb{E}}_{H \sim_p G_i}[|\textsc{Opt}(H)|] \\
&\leq \sum_{i=1}^{\ell} p\,|\textsc{Opt}(G_i)| \\
&= p\,|\textsc{Opt}(G)|.
\end{aligned}$$

For the first transition, $H \cap G_i$ is the graph with edges that are present in both H and G_i; the intuition behind this inequality is that we are essentially allowed to match the same vertices multiple times on the right hand side. The third and fourth transitions follow from Equations (12.3) and (12.4).

So, it remains to find a partition of G into edge-disjoint subgraphs, $G_1 \uplus G_2 \uplus \cdots \uplus G_\ell$, which satisfies (12.3) and (12.4). We prove that the Edmonds-Gallai Decomposition [115] can be used to construct a partition satisfying these properties.

Lemma 12.2.5 (Edmonds-Gallai Decomposition)**.** *Let $G = (V, E)$ be an undirected graph, let B be the set of vertices matched by every maximum cardinality matching in G, and let $D = V \setminus B$. Furthermore, partition B into subsets A and $C = B \setminus A$, where A is the set of vertices with at least one neighbor outside B. And let $D_1, \ldots, D_r$, be the connected components of the induced subgraph $G[D]$. Then the following properties hold.*

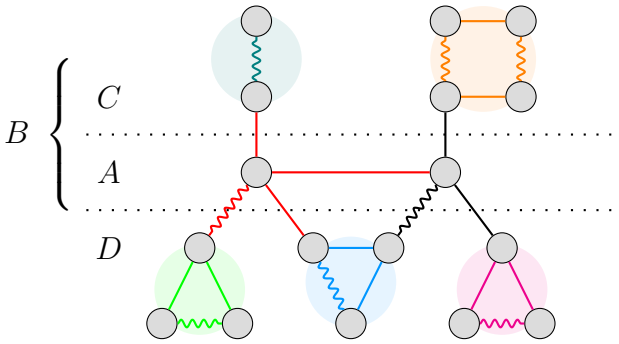


Figure 12.2: A graph demonstrating the Edmonds-Gallai Decomposition and the edge-disjoint graph partition for the proof of Lemma 12.2.3. In this graph, each color represents one G_i in the partition $G = \biguplus_i G_i$ and the wavy edges represent the matched edges in $\textsc{Opt}(G)$.

1. $\textsc{Opt}(G)$ *matches each node in* A *to a distinct connected component of* $G[D]$.

2. *Each* D_i *is* factor-critical*, i.e., deleting any one vertex of* D_i *leads to a perfect matching in the remainder of* D_i.

We now describe how the Edmonds-Gallai Decomposition is used to construct the desired partition of G. For the ith connected component of $G[C]$ and $G[D]$, create a subgraph G_i corresponding to its edges. Furthermore, for each vertex $i \in A$, create a subgraph G_i corresponding to the set of edges incident on i. If there is an edge between two vertices of A, i and i', then include the edge in only one of G_i or $G_{i'}$. Since the Edmonds-Gallai Decomposition has no edges between C and D, $G = \biguplus_i G_i$ forms a partition of the edge set of G. See Figure 12.2 for an example of this construction.

We argue that the foregoing partition satisfies Equation (12.3).

Claim 12.2.6. $|\textsc{Opt}(G)| = \sum_i |\textsc{Opt}(G_i)|$.

Proof. Let us define $\textsc{Opt}(G) \restriction^* G_i$ to be the vertices of $\textsc{Opt}(G) \restriction G_i$ that are matched by edges that lie within G_i. Since the decomposition is edge-disjoint, it holds that $|\textsc{Opt}(G)| = \sum_i |\textsc{Opt}(G) \restriction^* G_i|$. It is therefore sufficient to show that for all i, $|\textsc{Opt}(G_i)| = |\textsc{Opt}(G) \restriction^* G_i|$. There are three cases:

1. G_i corresponds to a component of $G[C]$. Recall that $\textsc{Opt}(G)$ matches all vertices of $C \subseteq B$. Moreover, C has no edges to D, and A is only matched with D, so the vertices of G_i have no matched edges outside G_i. It follows that $\textsc{Opt}(G) \restriction^* G_i$ is itself a perfect matching on G_i, and $|\textsc{Opt}(G_i)| = |\textsc{Opt}(G) \restriction^* G_i|$.

2. G_i corresponds to a star with vertex $i \in A$: For each $i \in A$, by the first property of the Edmonds-Gallai Decomposition, i is matched to a distinct component of $G[D]$. Therefore, $\textsc{Opt}(G) \restriction^* G_i$ includes an edge from $\textsc{Opt}(G)$. Since any star can have at most one matched edge, we have that $|\textsc{Opt}(G) \restriction^* G_i| = \textsc{Opt}(G)$.

3. G_i corresponds to a component of $G[D]$: Since such a component is factor-critical, it has an odd number of vertices, and, for any vertex, a maximum matching that covers all other vertices. Therefore, both $\textsc{Opt}(G_i)$ and $\textsc{Opt}(G) \restriction^* G_i$ match all but one vertex of this component, and $|\textsc{Opt}(G_i)| = |\textsc{Opt}(G) \restriction^* G_i|$.

□

Next, we show that $G = \biguplus_i G_i$ satisfies (12.4). There are three types of G_i in this partition: (i) G_i is a star, (ii) G_i is a component of $G[D]$ and has a matching that covers all but one vertex, and (iii) G_i is a component of $G[C]$ and has a perfect matching.

Let us first address case (i) — that of a star. Clearly it holds that $|\textsc{Opt}(G_i)| = 2$. Now, $|\textsc{Opt}(H)| \in \{0, 2\}$, and for $\textsc{Opt}(H)$ to be non-empty, H must include the center of the star, which happens with probability p.

The following claim establishes Equation (12.4) in cases (ii) and (iii). Note that this is the only place where the assumption $p \leq 1/2$ is used.

Claim 12.2.7. *For any $p \leq \frac{1}{2}$, and any graph G with n vertices such that $|\textsc{Opt}(G)| \geq n - 1$,*

$$\mathop{\mathbb{E}}_{H \sim_p G}[|\textsc{Opt}(H)|] \leq p\,|\textsc{Opt}(G)|.$$

Proof. Let $t \in \mathbb{N}$ and $p \in [0, 1/2]$. It holds that

$$\frac{1}{2} - \frac{1}{2}(1-2p)^{2t+1} \geq p, \tag{12.5}$$

because the left hand side is concave and has value 0 for $p = 0$ and $1/2$ for $p = 1/2$. We also use the equalities

$$\sum_{i=0}^{k} \binom{k}{i} x^i = (1+x)^k, \tag{12.6}$$

and

$$\sum_{i=1}^{k} i\binom{k}{i} x^{i-1} = k(1+x)^{k-1}. \tag{12.7}$$

Assume that $n = 2t+1$ for some $t \geq 0$. By the claim's assumption, it holds that $\textsc{Opt}(G) = 2t$. Any matching among a set of i vertices matches at most $2\lfloor i/2 \rfloor$ vertices. Hence, the expected matching size of the subgraph induced by a random set of vertices when each vertex is included

independently with probability p is

$$\begin{aligned}
\mathop{\mathbb{E}}_{H\sim_p G}[|\textsc{Opt}(H)|] &\leq \sum_{i=1}^{2t+1} 2\lfloor i/2\rfloor \binom{2t+1}{i} p^i (1-p)^{2t+1-i} \\
&= p(1-p)^{2t} \sum_{i=1}^{2t+1} i\binom{2t+1}{i} \left(\frac{p}{1-p}\right)^{i-1} \\
&\quad -\frac{1}{2}(1-p)^{2t+1} \sum_{i=0}^{2t+1} \binom{2t+1}{i} \left(\frac{p}{1-p}\right)^{i} \\
&\quad +\frac{1}{2}(1-p)^{2t+1} \sum_{i=0}^{2t+1} \binom{2t+1}{i} \left(\frac{p}{1-p}\right)^{i} (-1)^i \\
&= (2t+1)p - \frac{1}{2} + \frac{1}{2}(1-2p)^{2t+1} \\
&\leq 2tp,
\end{aligned}$$

where the penultimate transition follows by applying Equation (12.7) to the first term on the left hand side, and Equation (12.7) to the second and third terms; and the last transition follows from Equation (12.5).

If $n = 2t$ and $\textsc{Opt}(G) \geq n-1$, it must hold that $\textsc{Opt}(G) = 2t$, because each edge corresponds to two matched vertices. Moreover,

$$\begin{aligned}
\mathop{\mathbb{E}}_{H\sim_p G}[|\textsc{Opt}(H)|] &\leq \sum_{i=1}^{2t} 2\lfloor i/2\rfloor \binom{2t}{i} p^i (1-p)^{2t-i} \\
&\leq p(1-p)^{2t-1} \sum_{i=1}^{2t} i\binom{2t}{i} \left(\frac{p}{1-p}\right)^{i-1} \\
&= p(1-p)^{2t-1} 2t \left(1+\frac{p}{1-p}\right)^{2i-1} \\
&= 2tp.
\end{aligned}$$

Having established (12.4), the proof of Lemma 12.2.3 is now complete. □

12.2.2 Proof of Lemma 12.2.4

We will prove the following equivalent formulation of Lemma 12.2.4:

$$\Pr_{H\sim G}\left[|\textsc{Opt}(H)| \geq \mathop{\mathbb{E}}_{H\sim G}[|\textsc{Opt}(H)|] + \epsilon\right] \leq \exp\left(-\frac{\epsilon^2}{4L^2\ |\textsc{Opt}(G)|}\right), \tag{12.8}$$

and

$$\Pr_{H\sim G}\left[|\textsc{Opt}(H)| \leq \mathop{\mathbb{E}}_{H\sim G}|[\textsc{Opt}(H)|] - \epsilon\right] \leq \exp\left(-\frac{\epsilon^2}{2L^2\ \mathbb{E}[|\textsc{Opt}(H)|]}\right). \tag{12.9}$$

Let us first describe a failed approach for proving the lemma, which brings to light some subtleties in the above inequalities. Consider an explicit description of $|\textsc{Opt}(H)|$ as a function of n random variables, $X_1, \ldots, X_n$, where $X_i = 1$ if vertex i is in H and 0 otherwise. Then $|\textsc{Opt}(H)| = f(X_1, \ldots, X_n)$ is the size of the optimal matching on H. One can show that $f(\cdot)$ is L-Lipschitz, that is, changing X_i to $\neg X_i$, which corresponds to adding or removing one vertex from H, changes the size of the maximum matching by at most L. Lipschitz functions are known to enjoy strong concentration guarantees, as shown by McDiarmid's inequality,

$$\Pr\left[|f - \mathbb{E}[f]| > \epsilon\right] \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right),$$

where c_i is the Lipschitz constant for the ith variable, that is, for all i and for every possible input $x_1, \ldots, x_n$, $|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, \neg x_i, \ldots, x_n)| \leq c_i$.

While there are only $|\textsc{Opt}(G)|$ variables that truly participate in $\textsc{Opt}(G)$, even vertices that are not in $\textsc{Opt}(G)$ can participate in matchings of subsets of G, and as a result have a non-zero Lipschitz constant. Therefore, using McDiarmid's inequality for the concentration of $\textsc{Opt}(H)$ gives an $O(\sqrt{n})$ gap between $\textsc{Opt}(H)$ and $\mathbb{E}_{H\sim G}[\textsc{Opt}(H)]$.

Instead, in order to prove a gap of $\tilde{O}(\sqrt{|\textsc{Opt}(G)|})$, we use two alternative concentration bounds from statistical learning theory, which have recently been used to simplify and prove tight concentration and sample complexity results for learning combinatorial functions [265].

Lemma 12.2.8. [63, Theorem 12] *Let $X_1, X_2, \ldots, X_n$ be independent random variables, each taking values in a set $\mathcal{X}$. Let $f : \mathcal{X}^n \to \mathbb{R}$ be a measurable function. Let $X_1', \ldots, X_n'$ be independent copies of $X_1, \ldots, X_n$ and for all $i \in [n]$, define $f_i' = f(X_1, \ldots, X_i', \ldots, X_n)$. For all $i \in [n]$ and $\mathbf{x} \in \mathcal{X}^n$ assume that there exists $C > 0$, such that*

$$\mathbb{E}\left[\sum_{i=1}^{n} (f - f_i')^2 \cdot \mathbb{I}_{f > f_i'} \middle| \mathbf{x}\right] \leq C,$$

then for all $\epsilon > 0$,

$$\Pr[f > \mathbb{E}[f] + \epsilon] \leq e^{-\epsilon^2/4C}.$$

We show that the conditions of Lemma 12.2.8 hold for $f(x_1, \ldots, x_n) = \textsc{Opt}(H)$. Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{x}_i' = (x_1, \ldots x_i', \ldots, x_n)$. For all $\mathbf{x}$, let $H_{\mathbf{x}}$ be the subgraph corresponding to non-zero variables of $\mathbf{x}$. Note that if x_i is replaced by x_i' and the matching size is reduced, then the decrease is at most the maximum cycle length L. Furthermore, the only variables that can lead to a non-zero decrease from $|\textsc{Opt}(H_{\mathbf{x}})|$ to $|\textsc{Opt}(H_{\mathbf{x}_i'})|$ are variables that are in every optimal matching on $H_{\mathbf{x}}$. Therefore, there are at most $|\textsc{Opt}(H_{\mathbf{x}})| \leq |\textsc{Opt}(G)|$ such variables. We conclude that for all $\mathbf{x}$,

$$\mathbb{E}\left[\sum_{i=1}^{n} (f - f_i')^2 \cdot \mathbb{I}_{f > f_i'} \middle| \mathbf{x}\right] \leq L^2 \, |\textsc{Opt}(G)|.$$

The proof of the upper tail (12.8) follows immediately by using Lemma 12.2.8 with $C = L^2 \, |\textsc{Opt}(G)|$.

Unfortunately, Lemma 12.2.8 and its variants for lower-tail concentration cannot be used to establish the desired lower-tail bound (12.9). Indeed, consider the condition $\mathbb{E}[\sum_{i=1}^{n}(f - f_i')^2 \cdot \mathbb{I}_{f<f_i'}|\mathbf{x}] \le C$; while removing one of only $|\textsc{Opt}(G)|$ vertices can reduce the size of a matching, it may be possible that for some subgraph of G, adding any of the remaining vertices increases the size of the matching. Instead, we use the lower-tail concentration of self-bounding functions [64].

Definition 12.2.9. [64] *A function $g : \mathcal{X}^n \to \mathbb{R}$ is (a, b)-self-bounding if there exist functions $g_{-i} : \mathcal{X}^{n-1} \to \mathbb{R}$ for all $i \in [n]$ such that for all $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$ and $i \in [n]$,*

$$0 \le g(\mathbf{x}) - g_{-i}(\mathbf{x}_{-i}) \le 1,$$

and

$$\sum_{i=1}^{n} (g(\mathbf{x}) - g_{-i}(\mathbf{x}_{-i})) \le a \cdot g(\mathbf{x}) + b,$$

where $\mathbf{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ is obtained by dropping the ith component of $\mathbf{x}$.

Lemma 12.2.10. [64] *If $Z = g(X_1, \ldots, X_n)$, where $X_i \in \{0, 1\}$ are independent random variables and g is an (a, b)-self-bounding function with $a \ge \frac{1}{3}$, then for any $0 < t < \mathbb{E}[Z]$,*

$$\Pr[Z \le \mathbb{E}[Z] - t] \le \exp\left(-\frac{t^2}{2a \cdot \mathbb{E}[Z] + 2b}\right).$$

Let $g(\mathbf{x})$ be $\frac{1}{L}$ times the size of optimal matching on the subgraph whose vertices correspond to the non-zero x_i's, i.e., $g(\mathbf{x}) = \frac{1}{L}|\textsc{Opt}(H_{\mathbf{x}})|$. Define $g_{-i}(\mathbf{x}_{-i}) = \min_{x_i} g(\mathbf{x})$. We show that $g(\cdot)$ is $(L, 0)$-self-bounding.

Since $g_{-i}(\mathbf{x}_{-i})$ is the matching size in $H_{\mathbf{x}_{-i}}$ and $|\textsc{Opt}(H_{\mathbf{x}})| \ge |\textsc{Opt}(H_{\mathbf{x}_{-i}})| \ge |\textsc{Opt}(H_{\mathbf{x}})| - L$, we have that $0 \le g(\mathbf{x}) - g_{-i}(\mathbf{x}_{-i}) \le 1$. Furthermore, $g(\mathbf{x}) - g_{-i}(\mathbf{x}_{-i})$ is non-zero only if vertex i was in every $\textsc{Opt}(H_{\mathbf{x}})$. Since there are at most $|\textsc{Opt}(H_{\mathbf{x}})|$ such variables, we have

$$\sum_{i=1}^{n} (g(\mathbf{x}) - g_{-i}(\mathbf{x}_{-i})) \le |\textsc{Opt}(H_{\mathbf{x}})| = L \cdot g(\mathbf{x}).$$

Because $L > \frac{1}{3}$, we can use Lemma 12.2.10 with $\epsilon = tL$, and obtain

$$\begin{aligned}\Pr\left[|\textsc{Opt}(H_i)| \le \mathbb{E}[|\textsc{Opt}(H_i)|] - \epsilon\right] &\le \exp\left(-\frac{(\epsilon/L)^2}{2L(\frac{1}{L}\,\mathbb{E}[|\textsc{Opt}(H_i)|])}\right) \\ &\le \exp\left(-\frac{\epsilon^2}{2L^2\,\mathbb{E}[|\textsc{Opt}(H_i)|]}\right).\end{aligned}$$

□

Discussion on Self-Bounding Functions One might wonder whether the existing upper-tail bound of self-bounding functions could be used similarly to achieve an improved upper bound of $L\sqrt{2 \cdot \mathbb{E}[|\textsc{Opt}(H_i)|] \ln \frac{2k}{\delta}}$ for Lemma 12.2.4 — that is, a bound that depends on $|\textsc{Opt}(H_i)|$ instead of $|\textsc{Opt}(G_i)|$. Here, we answer this question in the negative. The next lemma bounds the upper tail of (a, b)-self-bounding functions.

Lemma 12.2.11. [64] *If $Z = g(X_1, \ldots, X_n)$, where $X_i \in \{0, 1\}$ are independent random variables and g is an (a, b)-self-bounding function, then for any $0 < t < \mathbb{E}[Z]$,*

$$\Pr[Z \geq \mathbb{E}[Z] + t] \leq \exp\left(-\frac{t^2}{2a \cdot \mathbb{E}[Z] + 2b + 2ct}\right),$$

where $c = \max\{0, (3a - 1)/6\}$.

Note that for $a = L > \frac{1}{3}$, the additional $2ct$ term in the denominator causes the upper-tail bound to decay only as a simple exponential, and leads to significantly weaker concentration. Whether the upper-tail bound of self-bounding functions can be improved to remove this term is an open problem in probability theory, with the first bound appearing in the work of Boucheron et al. [62], and improved bounds due to McDiarmid and Reed [205] and Boucheron et al. [64]. Successfully removing the $2ct$ term from the denominator would improve the result stated in Theorem 12.2.2 from $O\left(\sqrt{|\textsc{Opt}(G)| \ln(k/\delta)}\right)$ to $O\left(\sqrt{p_i|\textsc{Opt}(G)| \ln(k/\delta)}\right)$, and Corollary 12.3.1 from $O\left(k\sqrt{|\textsc{Opt}(G)| \ln(k/\delta)}\right)$ to $O\left(\sqrt{k|\textsc{Opt}(G)| \ln(k/\delta)}\right)$.

12.3 Individually Rational Matchings that Are Almost Optimal

In this section, we provide a *simple and practical* mechanism for kidney exchange that guarantees individual rationality, and with high probability yields a matching that is optimal up to lower-order terms. In comparison to the results of Section 12.2, its disadvantage is that it requires modifying deployed matching mechanisms, which simply return some optimal matching — our mechanism selects a *specific* matching (which may be suboptimal). However, it is only a minor modification, and therefore has the potential to inform practice.

Let us first consider the case where $\textsc{Opt}(G)$ is restricted to 2-cycles. In this case we can represent G as an undirected graph, as in Section 12.2.1. Let $H_1, \ldots, H_k$ be the subgraphs corresponding to the players. Consider the following matching mechanism, $\mathcal{M}(H_1, \ldots, H_k)$: First, compute the matching $M = \bigcup_i \textsc{Opt}(H_i)$; then grow M to a globally maximum cardinality matching by repeatedly applying augmenting paths. While an augmenting path changes the structure of a matching by adding and removing edges, it strictly expands the set of matched vertices. Therefore, this mechanism leads to a maximum cardinality matching on G, with the property that $\mathcal{M}(H_1, \ldots, H_k) \supseteq \bigcup_i \textsc{Opt}(H_i)$ for all $i \in [k]$. That is, $\mathcal{M}$ is individually rational.

The performance of the above mechanism for 2-cycles holds even when the subgraphs owned by players are chosen adversarially, rather than through a random process. Furthermore, this mechanism enjoys the stronger guarantee that every vertex that is matched under $\textsc{Opt}(H_i)$ is also matched under $\mathcal{M}(H_1, \ldots, H_k)$. As we discussed earlier (see Figure 12.1), these strong guarantees are unattainable when cycles of length 3 are allowed. But in our model for randomly generating $H_1, \ldots, H_k$, there is a mechanism that is individually rational and almost optimal, as we show next.

Corollary 12.3.1. *Let G be a directed graph. Consider optimal matchings on G that are restricted to constant-length cycles. For all $i \in [k]$, let $p_i = 1/k$. Then there exists a mechanism $\mathcal{M}$ such that $\mathcal{M}(H_1, \dots, H_k)$ is individually rational, and for any $\delta > 0$,*

$$\Pr\left[|\mathcal{M}(H_1, \dots, H_k)| \geq |\textsc{Opt}(G)| - O\left(k\sqrt{|\textsc{Opt}(G)| \ln \frac{k}{\delta}}\right)\right] \geq 1 - \delta.$$

As advertised, the mechanism underlying Corollary 12.3.1 is very simple: *Choose an arbitrary optimal matching* $\textsc{Opt}(G)$*, independently of* $H_1, \dots, H_k$*. If for all players* $i \in [k]$ *we have* $|\textsc{Opt}(H_i)| \leq |\textsc{Opt}(G) \restriction H_i|$*, then* $\mathcal{M}(H_1, \dots, H_k) = \textsc{Opt}(G)$*. Else, let* $\mathcal{M}(H_1, \dots, H_k) = \bigcup_i \textsc{Opt}(H_i)$. We call this mechanism the *Veto* mechanism, as any player can veto the proposed optimal matching. Alternatively, we can let players defect if they wish, while allowing the remaining players to continue to work together; for our mathematical purposes this is the same as the Veto mechanism, but the latter interpretation may be even more appealing from a practical viewpoint.

In a nutshell, the idea is that because $|\textsc{Opt}(H_i)|$ is concentrated around its expectation by Lemma 12.2.4, if some player wants to veto the proposed matching then it is likely that $\mathbb{E}[\textsc{Opt}(H_i)]$ is close to $|\textsc{Opt}(G) \restriction H_i|$, which is tightly concentrated around $|\textsc{Opt}(G)|/k$ by Hoeffding's inequality. But due to symmetry, this is true for all players, so the players can obtain on their own almost what they can obtain by collaborating.

Proof of Corollary 12.3.1. Let $p = 1/k$. If

$$\mathop{\mathbb{E}}_{H_i \sim_p G}[\textsc{Opt}(H_i)] \leq p\,|\textsc{Opt}(G)| - (2L+1)\sqrt{|\textsc{Opt}(G) \ln \frac{2k}{\delta}}$$

then

$$\begin{aligned}
&\Pr_{H_i \sim G}[|\textsc{Opt}(H_i)| > |\textsc{Opt}(G) \restriction H_i|] \\
&\qquad \leq \Pr_{H_i \sim G}\Bigg[|\textsc{Opt}(H_i)| - |\textsc{Opt}(G) \restriction H_i| \\
&\qquad\qquad + \left(p\,|\textsc{Opt}(G)| - \mathbb{E}[\textsc{Opt}(H_i)] - (2L+1)\sqrt{|\textsc{Opt}(G)| \ln \frac{2k}{\delta}}\right) > 0\Bigg] \\
&\qquad \leq \Pr_{H_i \sim G}\left[|\textsc{Opt}(H_i)| - \mathbb{E}[\textsc{Opt}(H_i)] > 2L\sqrt{|\textsc{Opt}(G)| \ln \frac{2k}{\delta}}\right] \\
&\qquad\qquad + \Pr_{H_i \sim G}\left[p\,|\textsc{Opt}(G)| - |\textsc{Opt}(G) \restriction H_i| > \sqrt{|\textsc{Opt}(G)| \ln \frac{2k}{\delta}}\right] \\
&\qquad \leq \frac{\delta}{k},
\end{aligned}$$

where the last inequality holds by the upper-tail bound of Lemma 12.2.4 and Hoeffding's inequality. Therefore, with probability $1 - \delta$, no player vetoes the proposed optimal matching, and $\mathcal{M}(H_1, \dots, H_k) = \textsc{Opt}(G)$ is optimal.

On the other hand, if

$$\mathop{\mathbb{E}}_{H_i \sim_p G}[\textsc{Opt}(H_i)] \geq p\,|\textsc{Opt}(G)| - (2L+1)\sqrt{|\textsc{Opt}(G)\ln\frac{2k}{\delta}},$$

then the expected total size of the internal matching is large, and we can fall back to the internal matchings. Indeed, note that $\mathbb{E}_{H_i \sim_p G}[|\textsc{Opt}(H_i)|] \leq p|\textsc{Opt}(G)|$ by symmetry. By the lower-tail bound of Lemma 12.2.4, with probability $1-\delta$, for all $i \in [k]$,

$$|\textsc{Opt}(H_i)| \geq \mathop{\mathbb{E}}_{H_i \sim_p G}[|\textsc{Opt}(H_i)|] - L\sqrt{2p|\textsc{Opt}(G)|\ln\frac{2k}{\delta}}.$$

Therefore, with probability $1-\delta$,

$$\begin{aligned}\sum_{i=1}^{k}|\textsc{Opt}(H_i)| &\geq k\mathop{\mathbb{E}}_{H_i \sim G}[|\textsc{Opt}(H_i)|] - kL\sqrt{\frac{2}{k}|\textsc{Opt}(G)|\ln\frac{2k}{\delta}}\\ &\geq |\textsc{Opt}(G)| - k(2L+1)\sqrt{|\textsc{Opt}(G)\ln\frac{2k}{\delta}} - kL\sqrt{\frac{2}{k}|\textsc{Opt}(G)|\ln\frac{2k}{\delta}}\\ &= |\textsc{Opt}(G)| - O\left(k\sqrt{|\textsc{Opt}(G)|\ln\frac{k}{\delta}}\right).\end{aligned}$$

So, $\mathcal{M}(H_1,\ldots,H_k) = \bigcup_i \textsc{Opt}(H_i)$ is a near optimal. □

We remark — without proof — that Corollary 12.3.1 still holds even if the probabilities, instead of being equal, are of the form $1/s^{t_i}$ for a fixed $s \in \mathbb{N}$, and possibly different $t_1,\ldots,t_n$. Similarly to Theorem 12.2.2, we conjecture that the statement actually holds for any $p_1,\ldots,p_n$ such that $p_i \geq 1/2$ for all $i \in [k]$. To prove this, one would need to strengthen Lemma 12.2.3, as discussed in Section 12.2. But now there is another difficulty: One would need to show that if $\mathbb{E}[\textsc{Opt}(H_i)]$ is close to $p_i|\textsc{Opt}(G)|$ for one player, then the same is true for all players — in which case falling back to the internal matchings is almost optimal. In the symmetric case, this claim trivially holds, which is precisely why we assume that $p_i = 1/k$ for all $i \in [k]$.

While we require a relatively strong assumption on the probabilities, it is satisfying that the theorem's bound is asymptotically tight. To show this, we present and analyze an example of a graph with n vertices where, with constant probability, *any* individually rational matching is smaller than the optimal matching by $\Omega(\sqrt{n})$.

Example 12.3.2. *Suppose that there are two players, each with probability $p = \frac{1}{2}$. Consider the graph G shown in Figure 12.3, which consists of four layers A, B, C and D, each with $n/4$ vertices. Any two layers of the graph are fully connected if there is an edge between them according to Figure 12.3. That is, the edge set of this graph is such that* any 3 vertices *from A, B, and C, respectively, form a directed 3-cycle, and* any 2 vertices *from C and D, respectively, form a directed 2-cycle. It is optimal to match the vertices in A, B, and C via 3-cycles, and therefore $|\textsc{Opt}(G)| = 3n/4$.*

It is easy to show, using the standard deviation of the binomial distribution, that the number of vertices a player owns in each layer deviates by $\pm\Theta(\sqrt{n})$ from its expectation (either larger

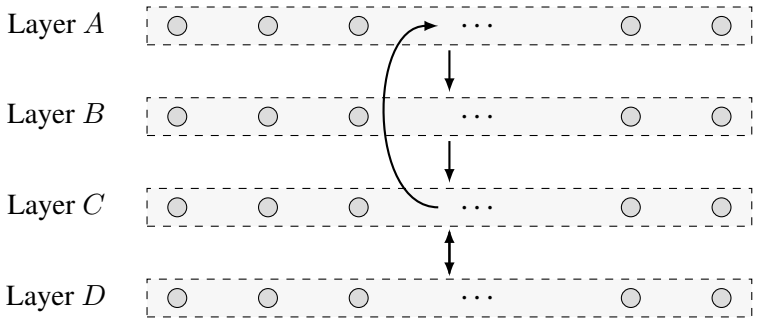


Figure 12.3: The graph construction of Example 12.3.2.

or smaller) with constant probability. Denote the number of vertices owned by player 1 *in layers A, B, C, and D by* a*,* b*,* c*, and* d*, respectively. We focus on the case where* $c \in \{n/8 + \sqrt{n}, \ldots, n/8 + (3/2)\sqrt{n}\}$*,* a *and* b *are both in* $\{n/8 - (3/2)\sqrt{n}, \ldots, n/8 - \sqrt{n}\}$*, and* $d \geq 3\sqrt{n}$ *— which happens with constant probability. Intuitively, player* 1 *is doing well, because he owns significantly more than half of the vertices in layer* C*, which is especially important.*

In the foregoing case, $\textsc{Opt}(H_1)$ *is obtained by taking* $\min\{a, b\}$ *3-cycles between* A*,* B*, and* C *(as many as possible), and then* $c - \min\{a, b\}$ *2-cycles between* D *and the unmatched vertices of* C*. Therefore,*

$$|\textsc{Opt}(H_1)| = 3 \cdot \min\{a, b\} + 2(c - \min\{a, b\}) = 2 \cdot c + \min\{a, b\} = 3n/8 + \Theta(\sqrt{n}).$$

On the other hand, consider some matching $\mathcal{M}$ *with* x *3-cycles and* y *2-cycles, such that (without loss of generality)* $x + y = n/4$ *— as the total number is constrained by the* $n/4$ *vertices in layer* C*. Note that under the optimal matching we have* $x = n/4$*, and*

$$|\textsc{Opt}(G) \restriction H_1| = a + b + c = 3n/8 - \Theta(n).$$

More generally, we have that $|\mathcal{M} \restriction H_1| \leq a + b + c + y$*. In order to guarantee that* $\mathcal{M}$ *is individually rational for player 1, we must close the gap between* $|\textsc{Opt}(H_1)|$ *and* $|\mathcal{M} \restriction H_1|$*, which implies that* $y = \Omega(\sqrt{n})$*. That is, we must sacrifice* $\Omega(\sqrt{n})$ *3-cycles in favor of 2-cycles. But that means that* $|\mathcal{M}| \leq |\textsc{Opt}(G)| - \Omega(\sqrt{n})$*.*

Finally, note that Corollary 12.3.1 assumes cycles of constant length (as does Theorem 12.2.2). As noted in Section 12.1, major kidney exchanges do, in fact, only use very short cycles and chains (which can also be represented as cycles) in each match run. But it is nevertheless interesting to point out that the same statement is false when long cycles are allowed. Indeed, in Section 12.4.2 we present an example of a graph with long cycles, where (with high probability) every individually rational matching is smaller than the optimal matching by $\Omega(n)$.

12.4 Justification for Conditions on p and L

In this section we present two examples that demonstrate that the requirement of $p < \frac{1}{2}$ and $L = O(1)$ are necessarily for Theorem 12.2.2.

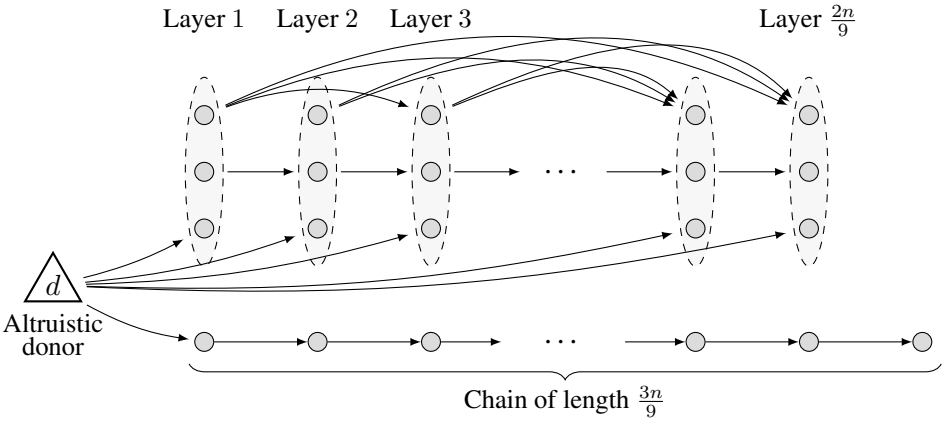


Figure 12.4: A graph demonstrating the problem with long cycles.

12.4.1 The Case of Large p

We first provide an example where the conclusion of Theorem 12.2.2 is violated under $p_i > 1/2$. This happens because the examples violate Lemma 12.2.3, that is, they satisfy

$$\mathop{\mathbb{E}}_{H\sim_p G}[|\textsc{Opt}(H)|] > \mathop{\mathbb{E}}_{H\sim_p G}[|\textsc{Opt}(G) \restriction H|] = p \cdot |\textsc{Opt}(G)|.$$

First, suppose that only 2-cycles are allowed. Consider a graph with three vertices v_1, v_2, v_3, and 2-cycles between v_1 and v_2, v_2 and v_3, and v_3 and v_1. Suppose $p = 2/3$. Then $p|\textsc{Opt}(G)| = 2 \cdot (2/3)$. On the other hand, $|\textsc{Opt}(H)| = 2$ if H contains at least two vertices (otherwise it is 0), hence

$$\mathop{\mathbb{E}}_{H\sim_p G}[|\textsc{Opt}(H)|] = 2\left(\binom{3}{2}p^2(1-p) + \binom{3}{3}p^3\right) = 2 \cdot \frac{20}{27} > 2 \cdot \frac{2}{3}.$$

Now consider a graph that contains many disjoint copies of the one just discussed. We have that both $\textsc{Opt}(H)$ and $|\textsc{Opt}(G) \restriction H|$ are concentrated around their expectations (by Hoeffding's inequality), so, with high probability, $|\textsc{Opt}(H)| > |\textsc{Opt}(G) \restriction H| + \Omega(n)$.

When 3-cycles are allowed too, it is possible to show that the same phenomenon happens, for a value of p sufficiently close to 1, in a graph with five vertices $v_1, \ldots, v_5$, and 2-cycles between v_i and v_{i+1} for $i = 1, \ldots, 4$, as well as between v_5 and v_1.

12.4.2 The Case of Long Cycles

We construct an example where long cycles are allowed, and every individually rational matching is smaller than the optimal matching by $\Omega(n)$. Motivated by kidney exchange, the example includes an altruistic donor, and matchings may include chains initiated by the altruist. However, we can easily transform the example into one where matchings can only include cycles, by adding directed edges from every vertex in the graph to the altruistic donor.

Consider the graph G in Figure 12.4. The altruistic donor d is shown as a triangle (we do not count him as one of the n vertices). The vertices consist of (i) a chain of $\frac{3n}{9}$ vertices, and (ii) a network of $\frac{2n}{9}$ layers — shown as dashed ellipses — with each layer consisting of three

vertices. All vertices in layer i have edges to all vertices in layers $(i+1), \ldots, \frac{2n}{9}$, and there are edges from d to all vertices in each layer. Observe that since there are no cycles in this graph, all matches must happen only via a chain that originates in d. The longest chain consists of $\frac{3n}{9}$ vertices. Thus, there is a unique optimal matching in G, and its size is $|\textsc{Opt}(G)| = \frac{3n}{9}$.

Let us now assume the there are two players with probability $p = \frac{1}{2}$ each. We first observe that the expected share of a subgraph $H \sim_p G$ in the optimal matching is $\mathbb{E}_{H \sim_p G}[|\textsc{Opt}(G) \restriction H|] = \frac{3n}{18}$.

Next we examine $\textsc{Opt}(H)$. We can assume that $d \in H$, as this is always true for one of the two players (so we focus on that player without loss of generality). For any given assignment of the other vertices to the players, we say that a layer is *good* if at least one of the vertices in that layer is in H. It is easy to see that — under the assumption of $d \in H$ — $\textsc{Opt}(H)$ is at least the number of good layers (via a chain that starts at d and visits each good layer in order). Notice that each layer is good with probability $1 - (1/2)^3 = 7/8$. Therefore, the expected number of layers that are good is $\frac{7}{8} \cdot \frac{2n}{9}$. It follows that $\mathbb{E}_{H \sim p}[|\textsc{Opt}(H)|] \geq \frac{14}{72}n$.

Since both $|\textsc{Opt}(H)|$ and $|\textsc{Opt}(G) \restriction H|$ are almost always within $O(\sqrt{n})$ of their expected values, we have that with high probability, $|\textsc{Opt}(H)| - |\textsc{Opt}(G) \restriction H| \geq \frac{2}{72}n - O(\sqrt{n})$. That is, $\textsc{Opt}(G)$ is not individually rational, and an individually rational matching would have to use d to initiate a chain into the layered network. But such a chain can have length at most $\frac{2n}{9}$, whereas $\textsc{Opt}(G) = \frac{3n}{9}$ — the difference is $\Omega(n)$, as desired.

12.5 Conclusions and Open Problems

Our work offers a new perspective on the design of optimal and individually rational matching mechanisms. Motivated by kidney exchange, we considered a setting where vertices of any graph (donor/patient pairs) are assigned at random to one of the players (hospitals). We showed that in any arbitrary graph, any optimal matching is *almost* individually rational, up to a lower order term of $O(\sqrt{|\textsc{Opt}(G)|})$. Furthermore, we introduced a simple matching mechanism that is fully individually rational and with high probability finds a matching that is within $O(\sqrt{|\textsc{Opt}(G)|})$ of the optimal matching.

From the theoretical perspective, two interesting extensions remain open. First, it would be interesting to extend these results to edge-weighted or vertex-weighted graphs. This is specifically motivated by kidney exchange, where edge weights represent the quality of a match between donors and patients, and vertex weights represent the urgency of the patients' health conditions and the expected improvement in their quality of life. Another interesting direction is to extend our results to a wider range of parameters p_i, which are induced by the size of each hospital, when the matching can consider any constant length cycles. It is not hard to see that Lemma 12.2.4, our concentration results, holds under both of these extensions. However, our proof of Lemma 12.2.3 relies on the structural properties the Edmonds-Gallai decomposition that do not immediately extend to edge-weighted and vertex-weights graphs or matchings with longer cycles. So, in order to address the above scenarios one only has to extend Lemma 12.2.3 to hold under these conditions.

Appendix A

Omitted Proofs for Chapter 5

A.1 Proof of Equation 5.4

Consider any vector of reserves $\mathbf{r} \in \mathcal{I}$ and let $\mathbf{r}' \in \mathcal{I}_m$ be the vector obtained by rounding each reserve price down to the nearest multiple of $1/m$, except for reserves that lie in the range $[0, 1/m)$, which are rounded up to $1/m$.

First it is easy to see that by rounding up all reserves in $\mathbf{r}$ that are below $1/m$, up to $1/m$, we cannot decrease the revenue by more than s/m. Let $\mathbf{r}''$ be this new vector. The reason is that any bidder that is now not allocated, could only have contributed a payment of at most $1/m$. Moreover, the revenue lost from other allocated bidders by doing this switch is at most $1/m$: the payment of another bidder i', changes only if his price setter was one of the bidders i that is no longer allocated, because he doesn't pass his new reserve r_i''. But then the payment of i' was at most $1/m$, since the value of i is at most $1/m$. Thus $\mathrm{Rev}(\mathbf{r}, \mathbf{v}) - \mathrm{Rev}(\mathbf{r}'', \mathbf{v}) \leq s/m$.

Now we assume that we start with a reserve price $\mathbf{r}$, such that $r_i \geq 1/m$ for all i, and let r' be the reserve vector where all reserves in $\mathbf{r}$ are rounded down to the nearest multiple of $1/m$. If $v_i > r_i$, then $v_i > r_i'$, so any bidder who would have been included in the basic VCG auction using reserves $\mathbf{r}$ is still included with $\mathbf{r}'$. This can only increase the number of bidders who are serviced and therefore pay a charge. We now show also that the total decrease in payment from bidders with value at least $1/m$ is at most s/m.

Consider the amount that serviced bidder i is charged. This is the maximum of r_i and the highest bid of a bidder in the basic VCG auction who was not serviced (or 0 if all bidders were serviced); let b denote this highest unserviced bid in the basic VCG auction under $\mathbf{r}$, and similarly let b' denote such a bid under $\mathbf{r}'$. Since the set of bidders entering the basic VCG auction increases from $\mathbf{r}$ to $\mathbf{r}'$, we must have $b' \geq b$.

Let U be the set of bidders serviced under $\mathbf{r}$, and U' the set under $\mathbf{r}'$. The difference in revenue is

$$\sum_{i \in U} \max\{r_i, b\} - \sum_{i \in U'} \max\{r_i', b'\}$$
$$= \sum_{i \in U \cap U'} (\max\{r_i, b\} - \max\{r_i', b'\}) + \sum_{i \in U \setminus U'} \max\{r_i, b\} - \sum_{i' \in U' \setminus U} \max\{r_{i'}', b'\}. \quad \text{(A.1)}$$

We begin by analyzing the last two terms. For any $i \in U \setminus U'$ and $i' \in U' \setminus U$,

$$r'_{i'} + 1/m > r_{i'} > v_{i'} \geq v_i > r_i,$$

where $v_{i'} \geq v_i$ follows, because i enters the basic VCG auction for $\mathbf{r}'$, but is not allocated the item. Therefore,

$$\max\{r'_{i'}, b'\} \geq \max\big\{r_i - 1/m,\, b\big\} \geq \max\{r_i, b\} - 1/m.$$

Since $|U \setminus U'| \leq |U' \setminus U|$, we can pick $V \subseteq U' \setminus U$ such that $|V| = |U \setminus U'|$ and obtain

$$\sum_{i \in U \setminus U'} \max\{r_i, b\} - \sum_{i' \in U' \setminus U} \max\{r'_{i'}, b'\} \leq \sum_{i \in U \setminus U'} \max\{r_i, b\} - \sum_{i' \in V} \max\{r'_{i'}, b'\} \leq \frac{|U \setminus U'|}{m} \quad .$$

Note that each term in the first sum of Equation (A.1) is at most $1/m$, because

$$\max\{r'_i, b'\} \geq \max\big\{r_i - 1/m,\, b\big\} \geq \max\{r_i, b\} - 1/m.$$

Thus, we have

$$\mathrm{Rev}(\mathbf{r}, \mathbf{v}) - \mathrm{Rev}(\mathbf{r}', \mathbf{v}) \leq \frac{|U \cap U'|}{m} + \frac{|U \setminus U'|}{m} \leq \frac{s}{m} \quad .$$

Thus if we start from any reserve price vector $\mathbf{r}$, we can first round up to $1/m$ all reserves that are below $1/m$ and then round down any other reserve to the nearest multiple of $1/m$. The combination of this two separate modification, can drop the revenue by at most $2s/m$. This yields the approximation result

$$\max_{\mathbf{r} \in \mathcal{I}} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{r}, \mathbf{v}_t) - \max_{\mathbf{r} \in \mathcal{I}_m} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{r}, \mathbf{v}_t) \leq \frac{2Ts}{m}. \tag{A.2}$$

A.2 Proof of Lemma 5.2.1

We first prove an approximate variant of Be-the-Leader Lemma.

Lemma A.2.1 (Be-the-Approximate-Leader Lemma)**.** *In the Generalized FTPL algorithm, for any $x \in \mathcal{X}$,*

$$\sum_{t=1}^{T} f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} \geq \sum_{t=1}^{T} f(x, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon T.$$

Proof. For $T = 1$ the inequality holds trivially, by the definition of $x_2 = \textsc{Opt}(x_1, \epsilon)$. Assume

that the claim holds for some T. Then, for all x

$$\begin{aligned}
\sum_{t=1}^{T+1} f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} &= \sum_{t=1}^{T} f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} + f(x_{T+2}, y_{T+1}) \\
&\geq \sum_{t=1}^{T} f(x_{T+2}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_{T+2}} - \epsilon T + f(x_{T+2}, y_{T+1}) \\
&\qquad \text{(by induction hypothesis)} \\
&= \sum_{t=1}^{T+1} f(x_{T+2}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_{T+2}} - \epsilon T \\
&\geq \sum_{t=1}^{T+1} f(x, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x} - \epsilon(T+1), \\
&\qquad \text{(by approximate optimality of } x_{T+2}\text{)}
\end{aligned}$$

proving the lemma. □

Proof of Lemma 5.2.1. Let $x^* = \arg\max_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t)$. Then by Lemma A.2.1,

$$\begin{aligned}
\textsc{Regret} &= \mathbb{E}\left[\sum_{t=1}^{T} f(x^*, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T} f(x^*, y_t) - \sum_{t=1}^{T} f(x_{t+1}, y_t)\right] + \mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right] \\
&\leq \mathbb{E}\left[\boldsymbol{\alpha} \cdot (\boldsymbol{\Gamma}_{x_1} - \boldsymbol{\Gamma}_{x^*})\right] + \mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right] + \epsilon T.
\end{aligned}$$

□

A.3 Proof of Lemma 5.3.9

Single-minded setting. In the single-minded setting, each bidder i is interested in one particular bundle of items $\hat{\mathbf{q}}_i$. That is, $v_i(\mathbf{q}_i) = v_i(\hat{\mathbf{q}}_i)$ for all $\mathbf{q}_i \supseteq \hat{\mathbf{q}}_i$ and 0 otherwise. Consider any vector of prices $\mathbf{a} \in \mathcal{P}$ and let $\mathbf{a}' \in \mathcal{P}_m$ be the vector obtained by rounding each price down to the nearest multiple of $1/m$, except any price below $1/m$, which is rounded up to $1/m$. First, it is easy to observe that by rounding item prices below $1/m$ up to $1/m$, can only lose $n \cdot k/m$ of revenue. In a second step, by then rounding all reserve prices down to the nearest multiple of $1/m$ can only lose an extra $n \cdot k/m$. Since the price of every bundle is reduced, any bidder i who received bundle $\hat{\mathbf{q}}_i$ in auction $\mathbf{a}$, also receives $\hat{\mathbf{q}}_i$ in auction $\mathbf{a}'$. So, the revenue of the mechanism reduces by at most $2nk/m$. That is,

$$\max_{\mathbf{a} \in \mathcal{P}} \sum_{t=1}^{T} \text{Rev}(\mathbf{a}, \mathbf{v}_t) - \max_{\mathbf{a} \in \mathcal{P}_m} \sum_{t=1}^{T} \text{Rev}(\mathbf{a}, \mathbf{v}_t) \leq \frac{2Tnk}{m}.$$

Unit-demand setting. In the unit-demand setting with infinite supply, each bidder i has $v_i(\mathbf{e}_\ell)$ for item ℓ, and wishes to purchase *at most one item*, i.e., item $\arg\max_\ell (v_i(\mathbf{e}_\ell) - a_\ell)$. We show that for any $\mathbf{a} \in \mathcal{P}$ there is $\mathbf{a}'' \in \mathcal{P}_m$ such that for any valuation profile $\mathbf{v}$, $\mathrm{Rev}(\mathbf{a}, \mathbf{v}) - \mathrm{Rev}(\mathbf{a}'', \mathbf{v}) \leq O(nk/m)$. At a high level, we first choose $\mathbf{a}'$, with $a'_\ell \in \{0, 1/m, \cdots, m/m\}$, as discounted prices such that items with higher prices are discounted at higher rates. It is not hard to see that under this condition, no bidder purchases a less expensive item in auction $\mathbf{a}'$. So, the loss in the revenue of the auction is bounded by the discount on the items. Using this intuition, it is sufficient to find $\mathbf{a}'$, with $a_\ell \in \{0, 1/m, \cdots, m/m\}$, such that $a_\ell \geq a'_\ell \geq a_\ell - O(1/m)$ for all $\ell \leq k$, and $a_\ell - a'_\ell > a_{\ell'} - a'_{\ell'}$ when $a_\ell > a_{\ell'}$. We then show how to derive $\mathbf{a}'' \in \mathcal{P}_m$ whose revenue is at least as good as $\mathbf{a}'$.

In the following, we provide one such mapping between $\mathbf{a}$ and $\mathbf{a}' \in \mathcal{P}_m$ that has an additive approximation guarantee. Hartline and Koltun [151] also provided a discretization of $\mathcal{P}$ that has multiplicative approximation guarantee, but using a discretized set different from $\mathcal{P}_m$.

Without loss of generality, assume $a_1 \leq a_2 \leq \cdots \leq a_k$. For ease of exposition, let $\epsilon = 1/m$. To begin, let a'_1 be the largest multiple of ϵ less than or equal to a_1. For $\ell = 2, \ldots, k$, let a'_ℓ be the largest multiple of ϵ less than or equal to a_ℓ such that $a_\ell - a'_\ell \geq a_{\ell-1} - a'_{\ell-1}$. Note that a'_ℓ is well defined, since we can begin by considering $a'_\ell = a'_{\ell-1}$ and then increase by ϵ until the condition is violated. This construction means that pricing of items with a larger ℓ is more attractive in $\mathbf{a}'$ than it was in $\mathbf{a}$. Thus, no bidder that prefers an item ℓ under $\mathbf{a}$, would prefer any item $\ell' < \ell$ under $\mathbf{a}'$. Therefore, the revenue obtained from the bidder i who prefers ℓ under $\mathbf{a}$ is at least a'_ℓ under $\mathbf{a}'$, which implies the bound

$$\mathrm{Rev}(\mathbf{a}, \mathbf{v}) - \mathrm{Rev}(\mathbf{a}', \mathbf{v}) \leq n \max_{\ell \leq k} (a_\ell - a'_\ell).$$

To complete the proof, we argue by induction that $a_\ell - a'_\ell \leq \ell\epsilon$. This clearly holds for $\ell = 1$. For $\ell \geq 2$, the definition of a'_ℓ, in the absence of discretization to ϵ, would yield $a'_\ell = a_\ell - (a_{\ell-1} - a'_{\ell-1})$. With the discretization, we have

$$a'_\ell \geq a_\ell - (a_{\ell-1} - a'_{\ell-1}) - \epsilon \geq a_\ell - (\ell - 1)\epsilon - \epsilon = a_\ell - \ell\epsilon,$$

where we used the inductive hypothesis at $\ell - 1$.

Next, we construct $\mathbf{a}''$, such that $a''_\ell = \epsilon$ if $a'_\ell = 0$ and $a''_\ell = a'_\ell$, otherwise. We show that any bidder i that purchased some item ℓ in auction $\mathbf{a}'$ with price $a'_\ell \geq \epsilon$, purchases item ℓ in auction $\mathbf{a}''$ as well.

Assume to the contrary that bidder i purchases another item ℓ'. Since there is infinite supply, bidder i could have purchased item ℓ' in auction $\mathbf{a}'$, but preferred to purchase item ℓ. Therefore,

$$v_i(\mathbf{e}_\ell) - a'_\ell \geq v_i(\mathbf{e}_{\ell'}) - a'_{\ell'}.$$

Note that $a''_\ell = a'_\ell$ and $a''_{\ell'} \geq a'_{\ell'}$ for all $\ell' \neq \ell$. So,

$$v_i(\mathbf{e}_\ell) - a''_\ell \geq v_i(\mathbf{e}_{\ell'}) - a''_{\ell'}.$$

Therefore, bidder i purchases item ℓ in auction $\mathbf{a}''$ as well.

Since only the bidder who receive items at price 0 may not be allocated the same items, we have that $\mathrm{Rev}(\mathbf{a}'', \mathbf{v}) \geq \mathrm{Rev}(\mathbf{a}', \mathbf{v})$. This completes the proof.

A.4 Proof of Lemma 5.4.1

Let $x^* = \arg\max_{x \in \mathcal{X}} \mathbb{E}_{y \sim F}[f(x, y)]$. By the definition of regret we have that for any $y_1, \ldots, y_T$,

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t}[f(x_t, y_t)] \geq \sup_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \textsc{Regret} \geq \sum_{t=1}^{T} f(x^*, y_t) - \textsc{Regret}. \tag{A.3}$$

Observe that the random variables $Z_t = f(x^*, y_t)$ are drawn i.i.d. with expected value $\mathbb{E}_{y \sim F}[f(x^*, y)]$ and are bounded in $[0, 1]$. Hence, by the Hoeffding bound, we get that with probability at least $1 - 2e^{-2T\epsilon^2}$:

$$\frac{1}{T} \sum_{t=1}^{T} f(x^*, y_t) \geq \mathop{\mathbb{E}}_{y \sim F}[f(x^*, y)] - \epsilon. \tag{A.4}$$

By setting $\epsilon = \sqrt{\frac{\log(2/\delta)}{2T}}$ and combining the two bounds we get the result.

A.5 Proof of Lemma 5.4.3

Let $x^* = \arg\max_{x \in \mathcal{X}} \mathbb{E}_{y \sim F}[f(x, y)]$. By the definition of regret we have that for any sequence $y_1, \ldots, y_T$,

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t}[f(x_t, y_t)] \geq \sup_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \textsc{Regret} \geq \sum_{t=1}^{T} f(x^*, y_t) - \textsc{Regret}. \tag{A.5}$$

Since $y_1, \ldots, y_T$ are a Markov chain, by applying Theorem 5.4.2 to this chain and to the function $f(x^*, \cdot)$, we obtain that with probability at least $1 - 2\exp\left(-\frac{T\gamma\epsilon^2}{4+10\epsilon}\right)$,

$$\frac{1}{T} \sum_{t=1}^{T} f(x^*, y_t) \geq \mathop{\mathbb{E}}_{y \sim F}[f(x^*, y)] - \epsilon. \tag{A.6}$$

If we set $\epsilon = \sqrt{\frac{14\log(2/\delta)}{\gamma T}}$ then we have, either $\epsilon > 1$, in which case the inequality is trivial, since $f(x, y) \in [0, 1]$, or if $\epsilon \leq 1$, then $\epsilon = \sqrt{\frac{14\log(2/\delta)}{\gamma T}} \geq \sqrt{\frac{(4+10\epsilon)\log(2/\delta)}{\gamma T}}$, which implies that the inequality holds with probability $1 - \delta$.

Appendix B

Omitted Proofs of Chapter 8

B.1 Proof of Lemma 8.5.2

For ease of presentation let $\eta = \frac{1-\beta}{2}$ be the probability of flipping the label in noisy regions. For $h_{\mathbf{w}^*}$ we have

$$L_\tau(h_{\mathbf{w}^*}) = 2\Big(\eta(\mathrm{dA} + \mathrm{dB}) + (1-\eta)(\mathrm{cA} + \mathrm{cB}) + \mathrm{cD}\Big).$$

For $h_{\mathbf{w}}$, note that area B relates to $h_{\mathbf{w}}$ as area D relates to $h_{\mathbf{w}^*}$ (and vice versa). Thus, the roles of B and D are exchanged for $h_{\mathbf{w}}$. We have

$$L_\tau(h_{\mathbf{w}}) = 2\Big(\eta(\mathrm{cA} + \mathrm{dD}) + (1-\eta)(\mathrm{dA} + \mathrm{cD}) + \mathrm{cB}\Big).$$

Hence,

$$L_\tau(h_{\mathbf{w}}) - L_\tau(h_{\mathbf{w}^*}) = 2\bigg((1-2\eta)(\mathrm{dA} - \mathrm{cA}) - \eta\Big((\mathrm{dB} - \mathrm{cB}) - (\mathrm{dD} - \mathrm{cD})\Big)\bigg).$$

Note that $\mathrm{dA} + \mathrm{dB} - \mathrm{dD} = \mathrm{dC}$ and $\mathrm{cA} + \mathrm{cB} - \mathrm{cD} = \mathrm{cC}$. Thus we get

$$\begin{aligned}
L_\tau(h_{\mathbf{w}}) - L_\tau(h_{\mathbf{w}^*}) &= 2\bigg((1-2\eta)(\mathrm{dA} - \mathrm{cA}) - \eta\Big((\mathrm{dB} - \mathrm{cB}) - (\mathrm{dD} - \mathrm{cD})\Big)\bigg) \\
&= 2\bigg((1-\eta)(\mathrm{dA} - \mathrm{cA}) - \eta\Big((\mathrm{dB} - \mathrm{cB}) + (\mathrm{dA} - \mathrm{cA}) - (\mathrm{dD} - \mathrm{cD})\Big)\bigg) \\
&= 2\Big((1-\eta)\big(\mathrm{dA} - \mathrm{cA}\big) - \eta\big(\mathrm{dC} - \mathrm{cC}\big)\Big) \\
&= \beta\big(\mathrm{dA} - \mathrm{cA}\big) - (1-\beta)\big(\mathrm{dC} - \mathrm{cC}\big).
\end{aligned}$$

B.2 Proof of Lemma 8.5.3

We have

$$\begin{aligned}\mathrm{dA} - \mathrm{cA} &= \frac{2}{\pi}\int_0^1 \int_0^\alpha \frac{z^2}{\tau}\sin(\varphi)\, d\varphi\, dz \\ &= \frac{2}{3\pi}\int_0^\alpha \frac{1}{\tau}\sin(\varphi)\, d\varphi \\ &= \frac{2}{3\pi\tau}[-\cos(\varphi)]_0^\alpha \\ &= \frac{2}{3\pi\tau}(1-\cos(\alpha)).\end{aligned}$$

B.3 Proof of Lemma 8.5.4

When $\tau \geq 1$,

$$\begin{aligned}\mathrm{dC} - \mathrm{cC} &= \frac{4}{\pi}\int_0^1 \int_{\frac{\pi-\alpha}{2}}^{\frac{\pi}{2}} \frac{z^2}{\tau}\sin(\varphi)\, d\varphi\, dz \\ &= \frac{4}{3\pi}\int_{\frac{\pi-\alpha}{2}}^{\frac{\pi}{2}} \frac{1}{\tau}\sin(\varphi)\, d\varphi \\ &= \frac{4}{3\pi\tau}\cos\left(\frac{\pi-\alpha}{2}\right) \\ &= \frac{4}{3\pi\tau}\sin\left(\frac{\alpha}{2}\right).\end{aligned}$$

For the case of $\tau < 1$, we have

$$\begin{aligned}\mathrm{dC} &= \frac{2}{\pi}\int_0^1 \int_{\frac{\pi-\alpha}{2}}^{\frac{\pi}{2}} z + \frac{z^2}{\tau}\sin(\varphi)\, d\varphi\, dz \\ &= \frac{\alpha}{2\pi} + \frac{2}{\pi}\int_0^1 \int_{\frac{\pi-\alpha}{2}}^{\frac{\pi}{2}} \frac{z^2}{\tau}\sin(\varphi)\, d\varphi\, dz \\ &= \frac{\alpha}{2\pi} + \frac{2}{3\tau\pi}\sin\left(\frac{\alpha}{2}\right).\end{aligned}$$

We now provide an upper bound on cC by integrating over a the triangular shape T (see Figure B.1). Note that this bound on cC is actually exact if $\tau \leq \cos(\alpha/2)$ and only a strict upper bound for $\cos(\alpha/2) < \tau < 1$. We have

$$\begin{aligned}\mathrm{cC} \leq (cT) &= \frac{2}{\pi}\cdot\int_0^\tau (1-\frac{z}{\tau})(z\tan(\alpha/2))\, dz \\ &= \frac{2}{\pi}\cdot\int_0^\tau z\tan(\alpha/2) - \frac{z^2}{\tau}\tan(\alpha/2)\, dz \\ &= \frac{\tau^2}{3\pi}\tan\left(\frac{\alpha}{2}\right).\end{aligned}$$

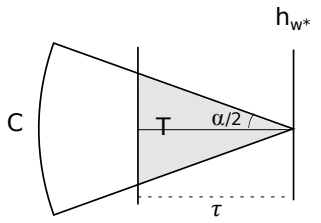


Figure B.1: Area T

Thus we get

$$(\mathrm{dC} - \mathrm{cC}) \geq (\mathrm{dC} - (cT)) = \frac{1}{\pi}\left(\frac{\alpha}{2} + \frac{2}{3\tau}\sin\left(\frac{\alpha}{2}\right) - \frac{\tau^2}{3}\tan\left(\frac{\alpha}{2}\right)\right).$$

Appendix C

Probability Lemmas for Chapter 9

Theorem C.0.1 (Probability of Ruin [119])**.** *Consider a player who starts with i dollars against an adversary that has N dollars. The player bets one dollar in each gamble, which he wins with probability p. The probability that the player ends up with no money at any point in the game is*

$$\frac{1-\left(\frac{p}{1-p}\right)^{N}}{1-\left(\frac{p}{1-p}\right)^{N+i}}.$$

Theorem C.0.2 (Bernstein Inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables with expectation μ. Supposed that for some positive real number L and every $k > 1$,*

$$\mathbb{E}[(X_i-\mu)^k] \leq \frac{1}{2}\,\mathbb{E}[(X_i-\mu)^2]L^{k-2}k!.$$

Then,

$$\Pr\left[\sum_{i=1}^{n} X_i - n\mu \geq 2t\sqrt{\sum_{i=1}^{n}\mathbb{E}[(X_i-\mu)^2]}\right] < \exp(-t^2), \quad \textit{for } 0 < t \leq \frac{1}{2L}\sqrt{\mathbb{E}[(X_i-\mu)^2]}.$$

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 1–9. ACM, 2004. 1

[2] Jacob Abernethy, Peter L Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal strategies and minimax lower bounds for online convex games. In *Proceedings of the 21st Conference on Computational Learning Theory (COLT)*, pages 415–424, 2008. 6.6

[3] D. J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Economics and Computation (EC)*, pages 295–304, 2007. 11.7.1, 11.9.1, 1

[4] Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters*, 111(15):731–737, 2011. 11.2.1, 12.1.3

[5] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1638–1646, 2014. 5.1

[6] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC)*, pages 355—255, 2014. 11.2.2, 12.1.2

[7] Nima Anari, Nika Haghtalab, Seffi (Joseph) Naor, Sebastian Pokutta, Mohit Singh, and Alfredo Torrico. Robust submodular maximization: Offline and online algorithms. *arXiv preprint arXiv:1710.04740*, 2017. 1.4

[8] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. A dynamic model of barter exchange. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1925–1933, 2015. 11.2.2

[9] Ross Anderson, Itai Ashlagi, David Gamarnik, and Alvin E Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112(3):663–668, 2015. 11.2.2, 11.8.2

[10] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016. 1

[11] Martin Anthony and Peter L Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. 1.1.2, 7.1.3, 7.3, 9.2, 10.2, 10.3.1, 10.5

[12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. 1.2

[13] Arash Asadpour, Hamid Nazerzadeh, and Amin Saberi. Stochastic submodular maximization. In *Proceedings of the 4th Conference on Web and Internet Economics (WINE)*, pages 477–489, 2008. 11.2.1

[14] Itai Ashlagi and Alvin E Roth. Free riding and participation in large scale, multi-hospital kidney exchange. *Theoretical Economics*, 9(3):817–863, 2014. 11.1, 11.8.2, 12.1, 12.1.2, 12.1.3, 2

[15] Itai Ashlagi, Duncan S. Gilchrist, Alvin E. Roth, and Michael Rees. Nonsimultaneous chains and dominos in kidney-paired donation—revisited. *American Journal of Transplantation*, 11(5):984–994, 2011. 11.1.2

[16] Itai Ashlagi, David Gamarnik, Michael A Rees, and Alvin E Roth. The need for (long) chains in kidney exchange, 2012. 11.8.1, 11.8.2, 12.1.2, 12.1.3

[17] Itai Ashlagi, Patrick Jaillet, and Vahideh H Manshadi. Kidney exchange in dynamic sparse heterogenous pools. In *Proceedings of the 14th ACM Conference on Economics and Computation (EC)*, pages 25–26, 2013. 11.8.1, 11.8.2

[18] Itai Ashlagi, Felix Fischer, Ian Kash, and Ariel D. Procaccia. Mix and match: A strategyproof mechanism for multi-hospital kidney exchange. *Games and Economic Behavior*, 91:284–296, 2015. 12.1.2, 12.1.3

[19] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 43–60. ACM, 2016. 11.2.3, 12.1.2

[20] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Symposium on Foundations of Computer Science (FOCS)*, pages 322–331, 1995. 4.1.2

[21] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The non-stochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002. 1.1.3

[22] Pranjal Awasthi and Tuomas Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 405–411, 2009. 11.2.2

[23] Pranjal Awasthi, Maria-Florina Balcan, and Philip M. Long. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 449–458. ACM, 2014. 8.1, 8.1.3

[24] Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Ruth Urner. Efficient learning of linear separators under bounded noise. In *Proceedings of the 28th Conference on Computational Learning Theory (COLT)*, pages 167–190, 2015. 1.2, 1.3, 8.1, 8.1.2, 8.1.3, 8.5, 9.1.2

[25] Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Hongyang Zhang. Learning and 1-bit compressed sensing under asymmetric noise. In *Proceedings of the 29th Conference on Computational Learning Theory (COLT)*, pages 152–192, 2016. 1.2, 1.3, 8.1, 8.1.1, 8.1.3, 8.5, 8.6.1, 9.1.2

[26] Pranjal Awasthi, Avrim Blum, Nika Haghtalab, and Yishay Mansour. Efficient PAC learning from the crowd. In *Proceedings of the 30th Conference on Computational Learning Theory (COLT)*, pages 127–150, 2017. 1.3

[27] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, February 2008. 1.1.3, 1.2, 4.1.2, 4.6.1, 4.6.3, 4.6.3, 5.1

[28] Yakov Babichenko and Aviad Rubinstein. Communication complexity of approximate nash equilibria. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 878–889. ACM, 2017. 1

[29] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Yaron Singer. Learning on a budget: posted price mechanisms for online procurement. In *Proceedings of the 13th ACM Conference on Economics and Computation (EC)*, pages 128–145. ACM, 2012. 9.1.2

[30] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks: Dynamic procurement for crowdsourcing. In *The 3rd Workshop on Social Computing and User Generated Content, co-located with ACM EC*, 2013. 9.1, 9.1.2

[31] Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *Proceedings of the 7th ACM Conference on Economics and Computation (EC)*, pages 29–35. ACM, 2006. 1.1.3, 5.1, 5.1.2, 5.1.3, 5.3.2, 5.5.1

[32] Maria-Florina Balcan and Vitaly Feldman. Statistical active learning algorithms. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, 2013. 8.1

[33] Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Proceedings of the 26th Conference on Computational Learning Theory (COLT)*, pages 288–316, 2013. 8.2

[34] Maria-Florina Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 65–72. ACM, 2006. 9.1.2

[35] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *Proceedings of the 20th Conference on Computational Learning Theory (COLT)*, pages 35–50, 2007. 8.1.2, 8.3.1

[36] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *Proceedings of the 25th Conference on Computational Learning Theory (COLT)*, pages 26.1–26.22, 2012. 10.1.2

[37] Maria-Florina Balcan, Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Commitment without regrets: Online learning in stackelberg security games. *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*, pages 61–78, 2015. 1.1.3, 1.3, 3.1.2

[38] Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 68:1–68:14, 2016. 1.4

[39] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Private and online optimization of piecewise lipschitz functions. *arXiv preprint arXiv:1212.2015*, 2017. 1

[40] Pierre Baldi and Søren Brunak. *Bioinformatics: the machine learning approach*. MIT press, 2001. 1

[41] Keith Ball, Eric A Carlen, and Elliott H Lieb. Sharp uniform convexity and smoothness inequalities for trace norms. *Inventiones mathematicae*, 115(1):463–482, 1994. 6.6

[42] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012. 11.2.1, 12.1.3

[43] Jonathan Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39, 1997. 10.1.2

[44] Jonathan Baxter. A model of inductive bias learning. *Journal of Artifificial Intellegience Research (JAIR)*, 12:149–198, 2000. 10.1.2

[45] Shai Ben-David and Reba Schuller. Exploiting task relatedness for mulitple task learning. In *Proceedings of the 16th Conference on Computational Learning Theory (COLT)*, pages 567–580, 2003. 10.1.2

[46] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010. 10.1.2

[47] Kshipra Bhawalkar and Tim Roughgarden. Welfare guarantees for combinatorial auctions with item bidding. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 700–709. SIAM, 2011. 5.6.2

[48] David Blackwell. Controlled random walks. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 336–338, 1954. 1.1.3

[49] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956. 1.1.3

[50] Avrim Blum. Machine learning: a tour through some favorite results, directions, and open problems. *FOCS 2003 tutorial slides, available at* `http://www.cs.cmu.edu/~avrim/Talks/FOCS03/tutorial.ppt`, 2003. 8.1.3

[51] Avrim Blum and Nika Haghtalab. Algorithms for generalized topic modeling. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 2730–2737, 2018. 1.4

[52] Avrim Blum and Jason D Hartline. Near-optimal online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1156–1163. SIAM, 2005. 1.1.3, 5.1

[53] Avrim Blum and Yishay Mansour. Learning, regret minimization, and equilibria. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 4. Cambridge University Press, 2007. 1.2, 4.3.1, 5.2

[54] Avrim Blum, A. Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1-2):35–52, 1998. 1.2, 8.1, 1, 8.1.3

[55] Avrim Blum, Anupam Gupta, Ariel D. Procaccia, and Ankit Sharma. Harnessing the power of two crossmatches. In *Proceedings of the 14th ACM Conference on Economics and Computation (EC)*, pages 123–140, 2013. 11.2.1, 12.1.2, 12.1.3

[56] Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Learning optimal commitment to overcome insecurity. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1826–1834, 2014. 1.3, 3.1.2, 4.1, 4.4, 4.4

[57] Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Lazy defenders are almost optimal against diligent attackers. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 573–579, 2014. 1.4

[58] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*, pages 325–342, 2015. 1.3, 11.2.3, 12.1.2

[59] Avrim Blum, Ioannis Caragiannis, Nika Haghtalab, Ariel D. Procaccia, Eviatar B. Procaccia, and Rohit Vaish. Opting into optimal matchings. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2351–2363. SIAM, 2017. 1.3

[60] Avrim Blum, Nika Haghtalab, Ariel D. Procaccia, and Mingda Qiao. Collaborative PAC learning. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2392–2401, 2017. 1.3

[61] Béla Bollobás. Random graphs. In *Modern graph theory*, pages 215–252. Springer, 1998. 11.1.1

[62] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. A sharp concentration inequality with applications. *Random Structures and Algorithms*, 16(3):277–292, 2000. 12.2.2

[63] Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures on Machine Learning*, pages 208–240. Springer, 2004. 12.2.8

[64] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. On concentration of self-bounding functions. *Electronic Journal of Probability*, 14(64):1884–1899, 2009. 12.1.2, 12.2.2, 12.2.9, 12.2.10, 12.2.11, 12.2.2

[65] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005. 1.2, 1.2, 8.1, 8.1.3, 8.2.1, 9.1.2

[66] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Sham Kakade. Towards minimax policies for online linear optimization with bandit feedback. In *Annual Conference on Learning Theory*, volume 23, pages 1–41, 2012. 1.1.3, 4.3.1

[67] Colin F Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2011. 1.2

[68] Ioannis Caragiannis, Aris Filos-Ratsikas, and Ariel D. Procaccia. An improved 2-agent kidney exchange mechanism. *Theoretical Computer Science*, 589:53–60, 2015. 12.1.2

[69] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. 10.1.2

[70] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006. 1.1.3, 6.1

[71] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004. 1.2, 5.1, 5.5

[72] Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2-3):321–352, 2007. 4.3.1

[73] Nicolo Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Regret minimization for reserve prices in second-price auctions. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1190–1204. SIAM, 2013. 1.1.3, 5.1

[74] Jeff Cheeger. *A lower bound for the smallest eigenvalue of the Laplacian*. 1969. 5.4.4

[75] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014. 1

[76] Jiecao Chen, Qin Zhang, and Yuan Zhou. Tight bounds for collaborative pac learning via multiplicative weights. *arXiv preprint arXiv:1805.09217*, 2018. 10.6

[77] Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 266–278, 2009. 11.2.1, 12.1.3

[78] Chao-Kai Chiang and Chi-Jen Lu. Online learning with queries. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 616–629, 2010. 6.2

[79] Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. Online optimization with gradual variations. In *Proceedings of the 25th Conference on Computational Learning Theory (COLT)*, pages 6–1, 2012. 6.2

[80] George Christodoulou, Annamária Kovács, and Michael Schapira. Bayesian combinatorial auctions. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 820–832. Springer, 2008. 5.6.2, 5.6.2

[81] Vincent Cohen-Addad and Varun Kanade. Online optimization of smoothed piecewise constant functions. In *Artificial Intelligence and Statistics*, pages 412–420, 2017. 7.1.4

[82] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994. 9.1.2

[83] Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 243–252. ACM, 2014. 5.1, 5.1.3

[84] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167. ACM, 2008. 1

[85] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Economics and Computation (EC)*, pages 82–90. ACM, 2006. 2.3

[86] Miguel Constantino, Xenia Klimentova, Ana Viana, and Abdur Rais. New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1):57–68, 2013. 11.8.2, 11.9.1

[87] Kevin P Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 822–833, 2012. 11.2.1, 12.1.3

[88] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. 8.1

[89] Amit Daniely. A PTAS for agnostically learning halfspaces. In *Proceedings of the 28th Conference on Computational Learning Theory (COLT)*, pages 484–502, 2015. 8.1.3

[90] Amit Daniely. Complexity theoretic limitations on learning halfspaces. pages 105–117, 2016. 8.1.3

[91] Partha Dasgupta, Peter Hammond, and Eric Maskin. The implementation of social choice rules: Some general results on incentive compatibility. *The Review of Economic Studies*, 46(2):185–216, 1979. 1

[92] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 235–242, 2005. 8.1.1, 9.1.2

[93] Constantinos Daskalakis and Christos H Papadimitriou. Three-player games are hard. In *Electronic colloquium on computational complexity*, volume 139, pages 81–87, 2005. 1

[94] Constantinos Daskalakis and Vasilis Syrgkanis. Learning in auctions: Regret is hard, envy is easy. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 219–228, 2016. 1.2, 4.9, 5.1, 2, **??**, 5.1.3, 5.2, 5.5, 5.6.2

[95] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. 1

[96] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979. 8.1

[97] Brian C Dean, Michel X Goemans, and Jan Vondrdk. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS)*, pages 208–217, 2004. 11.2.1

[98] Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *Proceedings of the 22nd Conference on Computational Learning Theory (COLT)*, 2009. 9.1, 9.1.2

[99] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 713–720, 2011. 10.1.2

[100] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13(1): 2655–2697, 2012. 8.1.3

[101] Ofer Dekel, Authur Flajolet, Nika Haghtalab, and Patrick Jaillet. Online learning with a hint. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, pages 5299–5308, 2017. 1.3

[102] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 1

[103] Nikhil R. Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. The sample complexity of auctions with side information. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 426–439. ACM, 2016. 5.1, 5.1.3, 5.4.2

[104] John P. Dickerson and Tuomas Sandholm. FutureMatch: Combining human value judgments and machine learning to match in dynamic environments. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 622–628, 2015. 11.2.2

[105] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Optimizing kidney exchange with transplant chains: Theory and reality. In *Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 711–718, 2012. 11.8.2, 12.1.2, 12.1.3

[106] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1340–1346, 2012. 11.2.2

[107] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. In *Proceedings of the 14th ACM Conference on Economics and Computation (EC)*, pages 323–340, 2013. 11.1.2, 11.2.2, 11.8, 11.8.1, 12.1.3

[108] John P. Dickerson, David F. Manlove, Benjamin Plaut, Tuomas Sandholm, and James Trimble. Position-indexed formulations for kidney exchange. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 25–42, 2016. 11.9.1

[109] Yichuan Ding, Dongdong Ge, Simai He, and Christopher Thomas Ryan. A non-asymptotic approach to analyzing kidney exchange graphs. In *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*, pages 257–258, 2015. 11.8.2

[110] Shahar Dobzinski and Noam Nisan. Mechanisms for multi-unit auctions. In *Proceedings of the 8th ACM Conference on Economics and Computation (EC)*, pages 346–351. ACM, 2007. 5.1.3, 5.6.1, 5.6.1, 5.6.1

[111] Dimitris C Dracopoulos and Simon Kent. Genetic programming for prediction and control. *Neural Computing & Applications*, 6(4):214–228, 1997. 1

[112] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. In *Proceedings of theAnnual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 169–178, 2011. 5.1

[113] Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 528–539, 2017. 1.1.3, 1.3, 4.9, 1, 5.1.3, 5.2.2, 5.1.1

[114] John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008. 8.1.3

[115] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. 12.1.2, 12.2.1

[116] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2589–2595, 2015. 1, 1.1.1, 3.1

[117] Uriel Feige, Yishay Mansour, and Robert Schapire. Learning and inference in the presence of corrupted inputs. In *Proceedings of the 28th Conference on Computational Learning Theory (COLT)*, pages 637–657, 2015. 8.1.3

[118] Michal Feldman, Hu Fu, Nick Gravin, and Brendan Lucier. Simultaneous auctions are (almost) efficient. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 201–210. ACM, 2013. 5.6.2, 5.6.2

[119] Willliam Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008. 9.4, C.0.1

[120] Peter J Fleming and Robin C Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control engineering practice*, 10(11):1223–1241, 2002. 1

[121] Mark French, Csaba Szepesvári, and Eric Rogers. *Performance of nonlinear approximate adaptive controllers*. John Wiley & Sons, 2003. 3.3.3

[122] Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the 22nd Conference on Computational Learning Theory (COLT)*, volume 90, pages 202–216, 1990. 9.4

[123] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995. 9.4, 10.6

[124] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997. 1.1.3, 1.2, 5.1, 7.3

[125] Komei Fukuda and Alain Prodon. Double description method revisited. In *Combinatorics and computer science*, pages 91–111. Springer, 1996. 2.4.1

[126] Martin Fürer and Huiwen Yu. Approximate the k-set packing problem by local improvements. *CoRR*, abs/1307.2262, 2013. 11.9

[127] Peter Gács and Laszlo Lovász. Khachiyan's algorithm for linear programming. In *Mathematical Programming at Oberwolfach*, pages 61–68. Springer, 1981. 2.4.4

[128] Walter Gautschi. On inverses of vandermonde and confluent vandermonde matrices. *Numerische Mathematik*, 4(1):117–123, 1962. 3.3.2

[129] Kristiaan M. Glorie, J. Joris van de Klundert, and Albert P. M. Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management*, 16(4):498–512, 2014. 11.8.2

[130] Gagan Goel and Pushkar Tripathi. Matching with our eyes closed. In *Proceedings of the 53rd Symposium on Foundations of Computer Science (FOCS)*, pages 718–727, 2012. 11.2.1, 12.1.3

[131] Sally A Goldman, Michael J Kearns, and Robert E Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM Journal on Computing*, 22(4):705–726, 1993. 5.1, 5.6.5, 5.6.3

[132] Michael F Goodchild and J Alan Glennon. Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth*, 3(3): 231–241, 2010. 1

[133] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 2nd edition, 1993. 2.4

[134] Theodore Groves, Roy Radner, and Stanley Reiter. *Information, Incentives, and Economic Mechanisms: Essays in Honor of Leonid Hurwicz*. U of Minnesota Press, 1987. 1

[135] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396. IEEE, 2017. 1

[136] Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Proceedings of the 16th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 205–216, 2013. 11.2.1, 12.1.3

[137] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R Ravi. Approximation algorithms for stochastic orienteering. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1522–1538, 2012. 11.2.1

[138] Rishi Gupta and Tim Roughgarden. A pac approach to application-specific algorithm selection. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 123–134. ACM, 2016. 7.1.4

[139] Rishi Gupta and Tim Roughgarden. A pac approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017. 1

[140] Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS)*, pages 742–765, 2006. 1.2

[141] Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1164–1173. SIAM, 2005. 5.1.2, 5.3.2, 5.3.2

[142] Nika Haghtalab, Fei Fang, Thanh Hong Nguyen, Arunesh Sinha, Ariel D. Procaccia, and Milind Tambe. Three strategies to success: Learning adversary models in security games. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 308–314, 2016. 1, 1.3

[143] Nika Haghtalab, Simon Mackenzie, Ariel D. Procaccia, Oren Salzman, and Siddhartha S. Srinivasa. The provable virtue of laziness in motion planning. *arXiv preprint arXiv:1710.04101*, 2017. 1.4

[144] Nika Haghtalab, Ritesh Noothigattu, and Ariel D. Procaccia. Weighted voting via no-regret learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1055–1062, 2018. 1.4

[145] Chen Hajaj, John P Dickerson, Avinatan Hassidim, Tuomas Sandholm, and David Sarne. Strategy-proof and efficient kidney exchange using a credit mechanism. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 921–928, 2015. 12.1.3

[146] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957. 1.1.3

[147] S. Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 353–360, 2007. 8.1.1

[148] S. Hanneke. Rates of convergence in active learning. *The Annals of Statistics*, 39(1): 333–361, 2011. 9.1.2

[149] Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014. ISSN 1935-8237. doi: 10.1561/2200000037. 8.1.1

[150] Jason D Hartline. Mechanism design and approximation. *Book draft. October*, 122, 2013. 5.4.2

[151] Jason D Hartline and Vladlen Koltun. Near-optimal pricing in near-linear time. In *Workshop on Algorithms and Data Structures*, pages 422–431. Springer, 2005. A.3

[152] Jason D Hartline and Tim Roughgarden. Simple versus optimal mechanisms. In *Proceedings of the 10th ACM Conference on Economics and Computation (EC)*, pages 225–234. ACM, 2009. 5.1.2, 5.3.1, 5

[153] Elad Hazan. The convex optimization approach to regret minimization. *Optimization for machine learning*, pages 287–303, 2012. 6.1

[154] Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *Proceedings of the 23th Conference on Computational Learning Theory (COLT)*, pages 165–188, 2008. 6.5

[155] Elad Hazan and Satyen Kale. Online submodular minimization. *Journal of Machine Learning Research*, 13(Oct):2903–2922, 2012. 1.2, 5.1, 5.2

[156] Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 128–141. ACM, 2016. 1.2, 5.1

[157] Elad Hazan and Nimrod Megiddo. Online learning with prior knowledge. In *Proceedings of the 20th Conference on Computational Learning Theory (COLT)*, pages 499–513. Springer, 2007. 6.2, 6.7.1

[158] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007. 6.1, 6.3, 6.3.2

[159] Xiuli He, Ashutosh Prasad, Suresh P. Sethi, and Genaro J. Gutierrez. A survey of stackelberg differential game models in supply and marketing channels. *Journal of Systems Science and Systems Engineering*, 16(4):385–413, Dec 2007. 1.1.1

[160] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 534–542, 2013. 9.1, 9.1.2

[161] Cor A. J. Hurkens and Alexander Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Computing*, 2(1):68–72, 1989. 11.1.1, 11.7.1, 11.7.2, 11.9.1

[162] Leonid Hurwicz. Optimality and informational efficiency in resource allocation processes. *Mathematical methods in the social sciences*, 1960. 1

[163] Marcus Hutter and Jan Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, 2005. 2, 5.2

[164] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 64–67. ACM, 2010. 1.2, 9.1

[165] Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Random graphs*, volume 45. John Wiley & Sons, 2011. 9.5

[166] Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5):1449–1475, 2003. 1.2

[167] Sham Kakade and Adam Tauman Kalai. From batch to transductive online learning. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 611–618, 2005. 1.2, 5.1

[168] Sham M Kakade, Adam Tauman Kalai, and Katrina Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1088–1106, 2009. 5.5

[169] Adam Tauman Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005. 1.2, 4.1.2, 4.3.1, 5.1, 5.1.1, 5.1.1, 5.2, 5.2.1, 5.5.1, 5.5.1, 6.1, 7.3

[170] Adam Tauman Kalai and Santosh Vempala. Simulated annealing for convex optimization. *Mathematics of Operations Research*, 31(2):253–266, 2006. 2.1, 2.3, 2.3.1, 2.4.2

[171] Adam Tauman Kalai, Adam R Klivans, Yishay Mansour, and Rocco A Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008. 8.1, 8.1.2, 8.1.3, 8.3, 5, 8.3.1, 8.3.2, 8.3.2

[172] Sampath Kannan, Jamie Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. *arXiv preprint arXiv:1801.03423*, 2018. 7.1.4

[173] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1953–1961, 2011. 9.1.2

[174] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014. 9.1, 9.1.2

[175] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015. 1

[176] Michael Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, 1994. 1.2, 8.1.3

[177] Michael Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Mach. Learn.*, 17(2-3), November 1994. 8.1.3

[178] Christopher Kiekintveld, Janusz Marecki, and Milind Tambe. Approximation methods for infinite Bayesian Stackelberg games: Modeling distributional payoff uncertainty. In *Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1005–1012, 2011. 2.1, 4.8

[179] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008. 1.2, 9.1

[180] Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 594–605. IEEE, 2003. 5.1

[181] Adam R. Klivans, Philip M. Long, and Rocco A. Servedio. Learning halfspaces with malicious noise. *Journal of Machine Learning Research*, 10:2715–2740, 2009. 8.1, 8.4

[182] Vladimir Koltchinskii. Rademacher complexities and bounding the excess risk in active learning. *Journal of Machine Learning Research*, 11:2457–2485, 2010. 9.1.2

[183] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 805–810, 2010. 2.3, 2.5

[184] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artifificial Intellegience Research (JAIR)*, 41:297–327, 2011. 2.2

[185] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413. Springer, 1999. 1

[186] Abhishek Kumar and Hal Daumé III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1103–1110, 2012. 10.1.2

[187] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, José A Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, et al. Machine learning in bioinformatics. *Briefings in bioinformatics*, 7(1):86–112, 2006. 1

[188] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002. 1

[189] Ruthanne Leishman, Richard Formica, Kenneth Andreoni, John Friedewald, Elizabeth Sleeman, Catherine Monstello, Darren Stewart, and Tuomas Sandholm. The Organ Procurement and Transplantation Network (OPTN) Kidney Paired Donation Pilot Program (KPDPP): Review of current results. In *American Transplant Congress (ATC)*, 2013. Talk abstract. 11.1.2

[190] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT)*, pages 250–262, 2009. 1.2, 2.1, 2.1, 2.4, 3.1.2, 4.1

[191] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 1

[192] Fei-Fei Li, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007. 1

[193] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988. 1.2, 7.1

[194] Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, Technical report, University of California, Santa Cruz, 1986. 1.1.3

[195] Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS)*, pages 256–261. IEEE, 1989. 1.1.3

[196] Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994. 4.1.2, 7.3

[197] László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures and Algorithms*, 30(3):307–358, 2007. 8.2

[198] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005. 3.2

[199] David F Manlove and Gregg OMalley. Paired and altruistic kidney donation in the uk: Algorithms and experimentation. In *International Symposium on Experimental Algorithms*, pages 271–282. Springer, 2012. 11.2.2

[200] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999. 1

[201] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *Proceedings of the 22nd Conference on Computational Learning Theory (COLT)*, pages 19–30, 2009. 10.1.2

[202] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1041–1048, 2009. 10.1.2

[203] Janusz Marecki, Gerry Tesauro, and Richard Segal. Playing repeated Stackelberg games with unknown opponents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 821–828, 2012. 4.1, 4.8

[204] Pascal Massart and Élodie Nédélec. Risk bounds for statistical learning. *The Annals of Statistics*, pages 2326–2366, 2006. 8.1

[205] Colin McDiarmid and Bruce Reed. Concentration for self-bounding functions and an inequality of Talagrand. *Random Structures and Algorithms*, 29(4):549–557, 2006. 12.2.2

[206] Daniel L McFadden. Quantal choice analaysis: A survey. *Annals of Economic and Social Measurement*, 5(4):363–390, 1976. 3.2

[207] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995. 1.2

[208] H Brendan McMahan. Analysis techniques for adaptive online learning. *arXiv preprint arXiv:1403.3465*, 2014. 6.5

[209] Marco Molinaro and R. Ravi. The query-commit problem. *CoRR*, abs/1110.0990, 2011. 11.2.1

[210] Jamie H Morgenstern and Tim Roughgarden. On the pseudo-dimension of nearly optimal auctions. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 136–144, 2015. 5.1, 5.1.2, 5.1.3, 5.3.3, 5.4.2, 5.4.2, 5.4.2

[211] Theodore S Motzkin. The double description method, in contributions to the theory of games ii. *Annals of Mathematics Study*, 28, 1953. 2.4.1

[212] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1): 58–73, 1981. 1, 5.1, 5.1.2, 5.3.3, 5.4.2

[213] Thanh H Nguyen, Francesco M Delle Fave, Debarun Kar, Aravind S Lakshminarayanan, Amulya Yadav, Milind Tambe, Noa Agmon, Andrew J Plumptre, Margaret Driciru, Fred Wanyama, et al. Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *Proceedings of the 6th Conference on Decision and Game Theory for Security (GameSec)*, pages 170–191, 2015. 3.1

[214] Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 2013. 1.2, 3.1.1, 3.2

[215] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 129–140. ACM, 1999. 1

[216] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. *Journal of Artifificial Intellegience Research (JAIR)*, 29(1):19–47, May 2007. 5.1, 5.1.3

[217] Ida Norheim-Hagtun and Patrick Meier. Crowdsourcing for crisis mapping in haiti. *Innovations: Technology, Governance, Globalization*, 5(4):81–89, 2010. 1

[218] Christos H Papadimitriou and Tim Roughgarden. Computing equilibria in multi-player games. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 82–91, 2005. 1

[219] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 895–902, 2008. 2.3

[220] D. Paulin. Concentration inequalities for Markov chains by Marton couplings and spectral methods. *arXiv preprint arXiv:1212.2015*, 2012. 5.4.1, 5.4.2

[221] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller. Tuned models of peer assessment in moocs. *arXiv preprint arXiv:1307.2579*, 2013. 1

[222] Gilles Pisier. Martingales in banach spaces (in connection with type and cotype). *Manuscript., Course IHP, Feb*, pages 2–8, 2011. 6.3.1

[223] James Pita, Manish Jain, Fernando Ordónez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Using game theory for los angeles airport security. *AI magazine*, 30(1):43, 2009. 3.1

[224] James Pita, Manish Jain, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Robust solutions to Stackelberg games: Addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence*, 174(15):1142–1171, 2010. 2.1

[225] Massimiliano Pontil and Andreas Maurer. Excess risk bounds for multitask learning with trace norm regularization. In *Proceedings of the 26th Conference on Computational Learning Theory (COLT)*, pages 55–76, 2013. 10.1.2

[226] Alexander Rakhlin and Karthik Sridharan. Statistical learning theory and sequential prediction. *Lecture Notes in University of Pennsylvania*, 2012. URL `http://stat.wharton.upenn.edu/~rakhlin/book_draft.pdf`. 7.5.1

[227] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *Proceedings of the 25th Conference on Computational Learning Theory (COLT)*, pages 993–1019, 2013. 6.2, 6.7.1

[228] Alexander Rakhlin and Karthik Sridharan. Bistro: An efficient relaxation-based method for contextual bandits. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1977–1985, 2016. 5.1

[229] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Stochastic, constrained, and smoothed adversaries. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1764–1772, 2011. 7.1.4

[230] Sasha Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3066–3074, 2013. 6.2, 6.7.1

[231] Ronald L Rivest and Robert Sloan. A formal model of hierarchical concept-learning. *Information and Computation*, 114(1):88–114, 1994. 1.2, 8.1, 8.1.3, 9.1.1

[232] Alvin E. Roth, Tayfun Sönmez, and M Utku Ünver. Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004. 11.9.2

[233] Alvin E. Roth, Tayfun Sönmez, and M Utku Ünver. Pairwise kidney exchange. *Journal of Economic theory*, 125(2):151–188, 2005. 11.9.2, 12.1.2

[234] Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97(3):828–851, 2007. 11.1, 12.1.2

[235] Yannig Roth, F Petavy, and J Céré. The state of crowdsourcing in 2015. *eYeka Analyst Report*, 2015. 1

[236] Tim Roughgarden. *Selfish routing and the price of anarchy*, volume 174. MIT press Cambridge, 2005. 1

[237] Tim Roughgarden and Okke Schrijvers. Ironing in the dark. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 1–18. ACM, 2016. 5.1, 5.1.3

[238] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM*, 49 (2):236–259, 2002. 1

[239] Tim Roughgarden and Joshua R Wang. Minimizing regret with multiple reserves. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 601–616. ACM, 2016. 1.1.3, 5.1, 5.3.1, 5, 5.5.1

[240] Ariel Rubinstein. *Modeling bounded rationality*. MIT press, 1998. 1.2

[241] Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. *ACM SIGecom Exchanges*, 15(2):45–49, 2017. 1

[242] Susan L Saidman, Alvin E Roth, Tayfun Sönmez, M Utku Ünver, and Francis L Delmonico. Increasing the opportunity of live kidney donation by matching for two and three way exchanges. *Transplantation*, 81:773–782, 2006. 11.8, 11.8.1

[243] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990. 9.4, 9.4.1, 9.4, 9.4.1

[244] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002. 1

[245] Rocco Anthony Servedio. *Efficient algorithms in computational learning theory*. Harvard University, 2001. 8.1, 8.1.1, 8.4

[246] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 7.3

[247] Shai Shalev-Shwartz, Ohad Shamir, and Karthik Sridharan. Learning kernel-based halfspaces with the 0-1 loss. *SIAM Journal on Computing*, 40(6):1623–1646, December 2011. 8.1.3

[248] Adish Singla and Andreas Krause. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1167–1178. ACM, 2013. 9.1.2

[249] Arunesh Sinha, Debarun Kar, and Milind Tambe. Learning adversary behavior in security games: A pac model perspective. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 214–222, 2016. 3.1, 3.1.2, 3.2, 3.2

[250] Aleksandrs Slivkins and Jennifer Wortman Vaughan. Online decision making in crowdsourcing markets: Theoretical challenges. *ACM SIGecom Exchanges*, 12(2):4–23, 2014. 9.1

[251] Robert H Sloan. PAC learning, noise, and geometry. In *Learning and Geometry: Computational Approaches*, pages 21–41. Springer, 1996. 1.2, 8.1, 8.1.3

[252] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3): 385–463, 2004. 1.2, 7.1.1, 7.5

[253] Jacob Steinhardt, Gregory Valiant, and Moses Charikar. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 4439–4447, 2016. 9.1.2

[254] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E. Schapire. Efficient algorithms for adversarial contextual learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2159–2168, 2016. 5.1, 5.6.2

[255] Vasilis Syrgkanis, Haipeng Luo, Akshay Krishnamurthy, and Robert E Schapire. Improved regret bounds for oracle-based adversarial contextual bandits. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3135–3143, 2016. 5.1

[256] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011. 1, 1.1.1, 2.1, 2.5

[257] Panos Toulis and David C. Parkes. Design and analysis of multi-hospital kidney exchange mechanisms using random graphs. *Games and Economic Behavior*, 91:360–382, 2015. 12.1.2, 12.1.3, 2

[258] Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014. 9.1, 9.1.2

[259] A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32(1):135–166, 2004. 8.1.3

[260] M Utku Ünver. Dynamic kidney exchange. *The Review of Economic Studies*, 77(1): 372–414, 2010. 11.2.2

[261] U.S. Department of Health and Human Services. Organ procurement and transplantation network, 2016. URL http://optn.transplant.hrsa.gov. 11.1

[262] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1.2, 9.1, 10.1

[263] Vladimir N Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998. 1.1.2

[264] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015. 1.2, 7.1

[265] Jan Vondrák. A note on concentration of submodular functions. arXiv:1005.2791, 2010. 12.2.2

[266] V Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998. 1.1.3

[267] Paul Wais, Shivaram Lingamneni, Duncan Cook, Jason Fennell, Benjamin Goldenberg, Daniel Lubarov, David Marin, and Hari Simons. Towards building a high-quality workforce with mechanical turk. *Presented at the NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, pages 1–5, 2010. 1.2, 9.1

[268] Jialei Wang, Mladen Kolar, and Nathan Srerbo. Distributed multi-task learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 751–760, 2016. 10.1.2

[269] Songbai Yan and Chicheng Zhang. Revisiting perceptron: Efficient and label-optimal learning of halfspaces. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1056–1066, 2017. 8.6.1

[270] Songbai Yan, Kamalika Chaudhuri, and Tara Javidi. Active learning from imperfect labelers. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2128–2136, 2016. 9.1.2

[271] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 453–460, 2014. 3.1

[272] Chicheng Zhang. Efficient active learning of sparse halfspaces. In *Proceedings of the 31st Conference on Computational Learning Theory (COLT)*, pages 1856–1880, 2018. 8.6.1

[273] Chicheng Zhang and Kamalika Chaudhuri. Active learning from weak and strong labelers. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 703–711, 2015. 9.1.2

[274] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, 2003. 4.3.1, 6.1