LEARNING FROM COLLECTIVE PREFERENCES, BEHAVIOR, AND BELIEFS

Jennifer Wortman Vaughan

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2009

Michael Kearns
Supervisor of Dissertation

Jianbo Shi
Graduate Group Chairperson

Acknowledgments

Thanks...

First and foremost, to my advisor, Michael Kearns. At the risk of sounding cliche, Michael has been a near ideal advisor to me. He helped me cultivate a taste in research problems, taught me how to recognize and design good models and algorithms, and was a constant source of invaluable advice on navigating the academic world. Perhaps most importantly, he helped me to develop confidence in myself as a researcher — no easy feat, I'm sure. I feel truly lucky to have had the chance to learn so much from him at the start of my career, and I sincerely hope that we will remain both collaborators and friends for many years to come.

To my thesis committee, Sanjeev Khanna, Yishay Mansour, Fernando Pereira, and Ben Taskar, for their feedback and advice, and for being so supportive of me and of this work.

To Kevin Leyton-Brown, Eugene Nudelman, Yoav Shoham, and the rest of the Multiagent Group at Stanford circa 2003, for giving me my first opportunity to get involved in research. It was because of Kevin's encouragement and contagious enthusiasm for research that I decided to go on for the Ph.D., so the credit (or blame) for me being here in the first place should go to him.

To Eyal Even-Dar and (again) Yishay Mansour, for acting as informal mentors and always giving me valuable career advice, whether I wanted to hear it or not.

To my other collaborators, colleagues, and teachers, from whom I have learned so much. I am especially grateful to Nina Balcan, John Blitzer, Yiling Chen, Koby Crammer, Mark Dredze, Lance Fortnow, Kuzman Ganchev, Amy Greenwald, Steve Hanneke, Stephen Judd, Sampath Kannan, Alex Kulesza, Nicolas Lambert, John Langford, Lihong Li, Dave Pennock, Dan Reeves, Lawrence Saul, Alex Strehl, Sid Suri, Jinsong Tan, Eugene Vorobeychik, Tong Zhang, and everyone who has

been a part of the mlunch crowd at Penn.

To the awesome staff who keep things running around the department. In particular, to Mike Felker, who has saved us all from bureaucratic disaster time and time again. The graduate program would collapse without him.

To Aaron, Aline, Debbie, Drew, Kathleen, Kristin, Maggie, Margaret, Nick, Rob, and all of the other people who have made Philadelphia feel like home to me over the past five years.

To Hanna Wallach, who talked me down from the academic ledge on more occasions than I can count. We are going to have a big drink when this is finished!

To my father, who got me hooked on logic puzzles as soon as I was old enough to read, and my mother, who always believed in me, and to the newer members of my family, Joan Linskey, Joanne and Mark Drexler, and Emily Vaughan, who warmly welcomed me into their lives and have been cheering me on ever since.

Finally, to Jeff, for putting up with my workaholic tendencies for the past five years, for believing I could do anything (and sometimes managing to make me believe too), and for being the best friend I've ever had, even through tough times. The best part about finishing this dissertation is knowing that we will go on to face what's next together. It's going to be great.

ABSTRACT

LEARNING FROM COLLECTIVE PREFERENCES, BEHAVIOR, AND BELIEFS

Jennifer Wortman Vaughan

Supervisor: Michael Kearns

Machine learning has become one of the most active and exciting areas of computer science research, in large part because of its wide-spread applicability to problems as diverse as natural language processing, speech recognition, spam detection, search, computer vision, gene discovery, medical diagnosis, and robotics. At the same time, the growing popularity of the Internet and social networking sites like Facebook has led to the availability of novel sources of data on the preferences, behavior, and beliefs of massive populations of users. Naturally, both researchers and engineers are eager to apply techniques from machine learning in order to aggregate and make sense of this wealth of collective information. However, traditional theories of learning fail to capture the complex issues that arise in such settings, and as a result, many of the techniques currently employed are ad hoc and not well understood.

The goal of this dissertation is to narrow this gap between theory and practice. To that end, we present a series of new learning models and algorithms designed to address and illuminate problems commonly faced when aggregating local information across large population. We build on the foundations of learning theory to examine the fundamental trade-offs that arise when aggregating preference data across many similar users to learn a model of a single user's tastes. We introduce and analyze a computational theory of learning from collective behavior, in which the goal of the algorithm is to accurately model and predict the future group behavior of a large population. We develop a forecaster that is guaranteed to perform reasonably well compared to best expert in a population but simultaneously never any worse than the average. Finally, we investigate the computational complexity of pricing in prediction markets, betting markets designed to aggregate individuals' opinions about the likelihood of future events, and propose an approximation technique based on the previously unexplored connection between prediction market prices and learning from expert advice.

Contents

Acknowledgments **ii**

1 Introduction **1**

1.1 Learning From Large Populations 2

1.1.1 Predicting Properties of the Population 2

1.1.2 Aggregating and Exploiting Individuals' Knowledge or Opinions 4

1.1.3 Topics This Dissertation Does Not Cover 6

1.1.4 Other Distinguishing Features of Collective Learning Problems 8

1.2 Overview of This Dissertation 8

1.2.1 Learning from Like-Minded Users 9

1.2.2 Learning from Collective Behavior 10

1.2.3 Aggregating Opinions Via Expert Advice 11

1.2.4 Aggregating Opinions Via Prediction Markets 12

1.3 Bibliographic Notes 13

2 Learning from Like-Minded Users **14**

2.1 Overview 15

2.2 The Learning Model 17

2.3 General Theory for the Multiple Source Problem 18

2.4 Simple Application to Binary Classification 22

2.5 Bounds Using Rademacher Complexity 24

2.5.1 Rademacher Complexity and General Lipschitz-Loss Bounds 24

2.5.2 Application to Classification Using Rademacher Complexity 25

2.5.3 Regression . . . 26
2.5.4 Remarks on the Use of Data-Dependent Complexity Measures . . . 27
2.6 Estimating the Disparity Matrix . . . 28
2.7 Synthetic Simulations . . . 29
2.8 A Few Words on Domain Adaptation . . . 32
2.8.1 A Bound Using Pairwise Divergence . . . 33
2.8.2 A Bound Using Combined Divergence . . . 34
2.9 Open Questions . . . 35

3 Learning from Collective Behavior 36

3.1 Overview . . . 37
3.2 The Model . . . 39
3.2.1 Agent Strategies and Collective Trajectories . . . 39
3.2.2 The Learning Model . . . 40
3.2.3 A No-Reset Variant . . . 42
3.2.4 Weaker Criteria for Learnability . . . 43
3.3 Social Strategy Classes . . . 44
3.3.1 Crowd Affinity: Mixture Strategies . . . 44
3.3.2 Crowd Affinity: Multiplicative Strategies . . . 46
3.3.3 Crowd Aversion and Other Variants . . . 47
3.3.4 Agent Affinity and Aversion Strategies . . . 48
3.3.5 Incorporating Network Structure . . . 49
3.4 A Reduction to I.I.D. Learning . . . 49
3.4.1 A Reduction for Deterministic Strategies . . . 50
3.4.2 A General Reduction . . . 51
3.5 Learning Social Strategy Classes . . . 53
3.5.1 Learning Crowd Affinity Mixture Models . . . 54
3.5.2 Learning Crowd Affinity Multiplicative Models . . . 58
3.6 Open Questions . . . 62

4 The Trade-Offs of Learning from Expert Advice **63**

4.1 Overview . . . 64

4.2 The Experts Framework . . . 68

4.3 The $\Theta(T)$ Frontier for Difference Algorithms . . . 69

4.3.1 The Difference Frontier Lower Bound . . . 70

4.3.2 A Difference Algorithm Achieving the Frontier . . . 71

4.4 Breaking the Difference Frontier . . . 72

4.4.1 Regret to the Best and Worst Experts . . . 73

4.4.2 PhasedAggression . . . 74

4.4.3 D-Prod . . . 76

4.5 Sketch of A General Lower Bound . . . 77

4.6 Open Questions . . . 78

5 Aggregating Opinions Via Prediction Markets and Machine Learning **79**

5.1 Overview . . . 80

5.2 Logarithmic Market Scoring Rules . . . 83

5.3 Complexity of Counting . . . 85

5.4 LMSR for Permutation Betting . . . 86

5.4.1 Subset Betting . . . 86

5.4.2 Pair Betting . . . 89

5.5 LMSR for Boolean Betting . . . 90

5.6 An Approximation Algorithm for Subset Betting . . . 92

5.6.1 Review of the Experts Setting . . . 92

5.6.2 Relationship to LMSR Markets . . . 93

5.6.3 Considering Permutations . . . 94

5.6.4 Approximating Subset Betting . . . 96

5.7 Open Questions . . . 99

6 Future Directions **100**

6.1 Improved Models for Collaborative Filtering . . . 100

6.2 Network Diffusion and Viral Marketing . . . 101

6.3 Social Search and Advertising . . . 102
6.4 Additional Connections Between Learning and Markets . . . 103

Appendix **104**

A1 Basic Tools from Probability Theory . . . 104
A1.1 Hoeffding's Inequality . . . 104
A1.2 McDiarmid's Inequality . . . 104
A2 Additional Proofs from Chapter 2 . . . 105
A2.1 Proof of Lemma 3 . . . 105
A2.2 Proof of Lemma 5 . . . 106
A2.3 Proof of Theorem 5 . . . 107
A2.4 Proof of Theorem 6 . . . 109
A3 Additional Proofs from Chapter 3 . . . 109
A3.1 Proof Sketch of Theorem 7 . . . 109
A3.2 Proof of Lemma 8 . . . 110
A3.3 Proof of Lemma 9 . . . 112
A3.4 Proof of Lemma 10 . . . 112
A3.5 Handling the case where Z_{a^*} is small . . . 112
A3.6 Proof of Lemma 12 . . . 115
A3.7 Bounding the $\mathcal{L}_1$ . . . 115
A3.8 Learning Without Resets . . . 116
A4 Additional Proofs from Chapter 4 . . . 118
A4.1 Proof of Theorem 16 . . . 118
A5 Additional Proofs from Chapter 5 . . . 126
A5.1 Proof of Theorem 18 . . . 126
A5.2 Proof of Theorem 19 . . . 128

References **130**

List of Tables

1.1 A goal-based characterization of problems one might wish to solve with data collected across large populations 3

4.1 Summary of the lower bounds presented in Chapter 4 66

4.2 Summary of the algorithmic results presented in Chapter 4 67

List of Figures

2.1 Visual illustration of Theorem 2 23
2.2 Simulation of the multiple source error bounds 30

3.1 Sample simulations of a) the crowd affinity mixture model, b) the crowd affinity multiplicative model, and c) the agent affinity model 46

4.1 The *BestWorst* algorithm 73
4.2 The *PhasedAggression* algorithm 74

A.1 The *GenerateBadSeq* procedure used in the proof of Theorem 16 119

Chapter 1

Introduction

Every day, web users flock to social networking sites like Facebook, personal data-sharing sites like Flickr, online prediction markets like Intrade, and massive e-commerce sites like Amazon. Today's web users are becoming increasingly comfortable sharing information about their interests and beliefs online, either as a way to keep in touch with family and friends or as a way to obtain more accurate personalized content in the form of search results, product recommendations, or advice. As a result, there is now a newfound wealth of available data, not only on the preferences, behaviors, and beliefs of huge populations of users, but also on the social links between members of these populations. While researchers and engineers are eager to apply machine learning techniques to this data, traditional theories of learning are not designed to handle the novel types of problems that arise. Not surprisingly, many of the systems currently in use are ad hoc.

This dissertation introduces a series of new models and algorithms designed to address the problems faced when aggregating local information across large populations of users. We build on the theoretical foundations of machine learning to provide answers to some of the fundamental questions that arise when learning from large populations:

- Suppose we want to learn a model of a single web user's likes and dislikes. Under what circumstances can we benefit from data on the preferences of other similar users?
- After observing the behavior of a large interacting population, can we accurately predict the population's future collective behavior?
- Suppose we would like to build a single forecaster to predict whether average stock prices

will rise or fall each day by aggregating the predictions of a population of experts. Can we build a single forecaster that is simultaneously guaranteed to perform "not too much worse" than the most accurate expert in the population and better than the population average?

- How can we efficiently aggregate the individual beliefs of a large population into a single accurate prediction about a future event with exponentially many possible outcomes?

Before examining these questions in more detail, we take a step back and provide a brief overview of the general types of learning problems one might be interested in solving using data collected across large populations and some of the distinguishing features of these problems.

1.1 Learning From Large Populations

There are a variety of natural ways one might choose to characterize the types of problems that researchers and engineers are interested in solving with access to data from large populations. Table 1.1 offers one such characterization based on five distinct high-level goals or agendas, including examples of problems or applications that attack each goal. We discuss each of these goals in turn below before describing some other distinguishing features of social learning problems in Section 1.1.4.

1.1.1 Predicting Properties of the Population

Perhaps the most natural goal from a machine learning perspective is to use data collected from the population to form predictions about the current or future state of the population. These predictions may pertain to individuals or to the population as a whole.

Consider the problem of predicting which websites a specific web user is likely to enjoy. We could attempt to solve this problem in isolation, considering only information about which websites the user has liked or disliked in the past. However, intuitively it should be possible to make more accurate predictions by incorporating website ratings contributed by other users. The problem of providing users with accurate recommendations about products they might like or dislike based on their own recorded preferences and the preferences of others is commonly known as *collaborative filtering* and is faced by companies such as Amazon and Netflix. In this problem, although we may

1. Predicting (local or global) properties of the population:
Collaborative filtering & preference modeling (Chapter 2)
Predicting the final outcome of an election from evolving voter preference data (Chapter 3)
2. Aggregating and exploiting the population's knowledge or beliefs:
Predicting the daily rise or fall of the stock market from expert advice (Chapter 4)
Predicting the final outcome of an election from individuals' inside knowledge (Chapter 5)
3. Helping the population perform (collective or individual) tasks:
Designing methods for users to aggregate personal data without violating privacy [81, 82]
Building reputation systems to help web users decide whom to trust [107, 3]
4. Manipulating the behavior of the population:
Incentivizing populations of advertisers to bid their true values for search ads [126, 50]
Maximizing the impact of a viral marketing campaign [108]
5. Determining how an individual in the population should behave:
Learning to obtain a high reward in multi-player stochastic games like online poker [129]
Learning to coordinate with others to achieve a common goal [34, 61]

Table 1.1: A goal-based characterization of the types of problems researchers and engineers might wish to solve with data collected across large populations, and examples of problems or applications that attack each goal.

only care about the preferences of one individual, it is the ability to aggregate data across the entire population that allows good models to be learned.

As another example, consider the problem of predicting the outcome of an election based on preference data collected over time (for example, via polls). While we might try to predict which candidate each individual voter will choose, one could argue that what really matters is only the ability to predict properties of the *collective* outcome of the election, such as the fraction of votes each candidate is likely to receive or the probability that any single candidate will secure at least half the votes.

Whether the goal is to make predictions about specific individuals or the population as a whole, there are two high level modeling approaches one might choose to use. The first option is to learn individual models for each user in the population. In the case of collaborative filtering, this corresponds to learning a separate target function for each user, a technique that is explored in more detail in Chapter 2. When predicting the outcome of an election, this corresponds to first learning a model of how each individual in the population is likely to behave, and subsequently combining the effects of this individual behavior to come up with a collective prediction. This technique is

discussed in Chapter 3.

The second option is to learn a single model that simultaneously captures the state of the population as a whole. For collaborative filtering, this is frequently done using low-rank matrix techniques [6, 117, 118]. In this framework, each known preference rating is entered into a matrix, with rows representing users and columns representing (say) websites. Missing entries are then approximated in such a way that the completed matrix has low rank, under the theory that only a small number of (unknown) factors influence whether or not a given user will enjoy a given website, and a user's rating of a website depends solely on how much that particular user cares about each of the factors. For election predictions, this might correspond to learning a single model to predict the outcome based on general properties or statistics of the population as a whole. Learning a single model has the advantage that it can sometimes be more efficient than learning individual models for each user, but there are situations in which a collection of individual models can be more powerful.

1.1.2 Aggregating and Exploiting Individuals' Knowledge or Opinions

Another natural goal that has received a lot of attention is aggregating the knowledge or advice of members of a population, for example to form predictions about the outcomes of future events. Variants of this problem have been studied in detail by two diverse and mostly disjoint communities, leading to two mature bodies of work of different flavors.

On the one hand, the extensive and still-growing literature on "no-regret learning" or "learning from expert advice" has established that on any sequence of T trials in which the predictions of a population of N individuals (referred to, perhaps misleadingly, as "experts") are observed, it is possible to maintain a dynamically weighted prediction whose cumulative performance (in terms of some kind of reward or loss) is within $O(\sqrt{T \log N})$ of the performance of the best single expert in hindsight (that is, after the full sequence has been revealed). This amazing guarantee holds even in a fully adversarial setting, when no distributional assumptions are made about the experts' performance; see Cesa-Bianchi and Lugosi [25] for a thorough overview of this topic.

While these results are extremely impressive, they are based on what we might call a "needle in a haystack" point of view. The guarantees have teeth only when we make the implicit assumption that there exist a small number of individuals in the population who dramatically outperform

the rest. When this is the case, the goal of the algorithms essentially boils down to tracking the performance of these superior experts.

Meanwhile, economists have been studying this problem from a different perspective, using a natural incentive scheme to entice individuals to contribute to global predictions via prediction markets [62, 63, 105]. A *prediction market* is a betting market designed to aggregate individual beliefs about the outcome of an event into a single prediction. A standard binary prediction market allows bets along a single dimension. For example, bettors might trade shares of a security that pays off $\$1$ if and only if Ford files for bankruptcy by the end of the year. If the current market price of a share is $\$p$, then a rational, risk-neutral bettor should be willing to buy shares if he believes the true probability that Ford will go bankrupt is greater than p. Conversely, he should be willing to sell shares at this price if he believes that the true probability is lower. The current price per share can be viewed as an estimate of how likely it is that Ford will go bankrupt this year according to the population as a whole. Studies have shown that the forecasts obtained through prediction markets are frequently more accurate than the predictions of individual domain specialists. For example, the price of orange juice futures is a better predictor of weather than the National Weather Service forecasts [109], while Oscar markets tend to be more accurate at predicting winners than expert columnists [106].

In contrast to the literature on learning from expert advice, we can think of the prediction market approach as encompassing a "wisdom of crowds" point of view. There is no longer any need to assume the existence of a small number of individuals who outperform the rest. The power comes instead from the fact that different individuals have access to different private information and therefore begin with a diverse set of beliefs.

In this thesis, we aim to get the best of both works. In Chapter 4, we explore what happens when we import the "wisdom of crowds" way of thinking into the expert advice setting, while in Chapter 5, we see how algorithms from the expert advice setting can be applied to pricing problems in the prediction market setting.

It is worth noting that other interpretations of the problem of harnessing the wisdom of crowds have been studied in the machine learning literature as well. One notable example is the recent line of work on supervised learning in settings in which individual, possibly malicious members of population provide the labeled examples to the learner [44, 45]. This work is directly applicable

to data collected via "crowdsourcing" websites such as Galaxy Zoo, where users are invited to label training images of galaxies, or Amazon's Mechanical Turk, in which individuals can receive a small payment for participating in a wide variety of crowdsourcing tasks.

1.1.3 Topics This Dissertation Does Not Cover

The remaining three high-level goals laid out in Table 1.1 are not addressed directly in this dissertation, but are worth elaborating on to obtain a more complete picture of problems that come up in social applications and collective learning as a whole.

First, one might wish to apply machine learning techniques in order to help a population perform a task, either collectively or individually using globally-aggregated information. For example, the website hunch.com helps individual users make decisions about every-day dilemmas such as which digital camera to purchase, whether or not to donate blood, or which weight-loss program to try by automatically aggregating advice from users with similar personality traits or desires. The collaborative filtering problem described above can also be viewed as serving this goal; collaborative filtering algorithms help users make better decisions about movies to watch, books to read, and websites to visit based on information gathered across the population. In a similar vein, the literature on reputation systems [107, 3] provides tools that help web users determine who in their population to trust.

An example of an application designed to help the population perform a *collective* task is the recently developed privacy-preserving belief propagation protocol [81, 82], which allows members of a social network to aggregate their private information so that each individual can learn relevant pieces of information (such as how likely it is that he has a contagious disease) without learning too much about anyone else on the network (such as which of his friends are likely to be infected).

There are many situations in which it is desirable to manipulate the behavior of a population. Consider the search advertising problem faced by search engines like Google and Yahoo [126, 50]. Large pools of advertisers place bids on various search terms with the hope of having their ads displayed. The set of ads displayed when a user searches for a given term is then determined by an auction mechanism. Advertisers pay the search engine a fee (which is also determined by the auction mechanism) only when their ads are clicked. In this example, the revenue of the search engine roughly grows as advertisers' bids increase, so it is in the best interest of the search engine

to motivate each advertiser to choose a bid per click that is close to the true value that the advertiser has for getting the click.

The search advertising problem is an example of the more general class of *mechanism design* problems [127, 104, 100]. At a high level, mechanism design refers to the subfield of game theory in which a designer is given control over the rules of the game being played. The designer can use this power to achieve his own objective. Prediction market design is another example. In this case, the goal of the designer is to design the rules of the betting market to encourage users to participate in the first place and to place their bets in such a way as to reveal useful information about their private beliefs. There have been some recent attempts to study the connection between mechanism design and machine learning [8], but this work is far outside the scope of this dissertation.

Interesting motivational problems also arise in settings in which the algorithm or designer is given little or no control over the rules that the population must obey or the rewards that they receive. For example, advertisers are interested in the problem of influence maximization for viral marketing [108]. Here the goal of the algorithm is to determine the optimal group of individuals in a social network to target with an advertising campaign in order to cause a new product or technology to spread virally throughout the network. Some open problems in this area are discussed in Chapter 6.

Finally, one might be interested in using machine learning techniques to figure out the optimal way for an individual to behave when interacting with other members of a large population. For example, when presented with polling data, instead of trying to make a global prediction about the outcome of the election as described above, we might instead choose to ask if there are ways in which one individual or small group of individuals can alter their actions or stated beliefs in order to influence the outcome of the election. In the search advertising setting, we might ask how individual advertisers can alter their bids in order to achieve a desired outcome, such as forcing their competitors to pay more per click. In other scenarios, we might ask instead how a collection of individuals can independently learn strategies that allow them to coordinate with each other to achieve a common goal [34, 61]. Variants of this type of problem have been examined in great detail in the vast literature on multiagent learning [70, 64, 115] and in the literature on reinforcement learning more generally [121].

1.1.4 Other Distinguishing Features of Collective Learning Problems

In addition to the different motivations and goals one might consider for learning across large populations, there are other distinguishing features of collective learning problems that are worthy of discussion. One obvious distinguishing feature is the amount of structure that exists in the population. In some applications, such as viral marketing and the privacy-preserving belief propagation protocol mentioned above, there is an explicit social network defined over individuals, representing friendships, collaborations, or other binary relationships between members of the population. In other problems, such as collaborative filtering, there is an implicit "soft" or weighted network over individuals, for example representing the level of similarity between pairs of people. Finally, in problems such as learning from expert advice, there is generally no need to assume any structure over the population at all.

Another distinguishing feature of these problems is how much (and in what way) the algorithm is able to observe the population. In collaborative filtering, it is assumed that each member of the population voluntarily provides the algorithm with a full description of his preferences over (say) websites with which he is already familiar. On the contrary, in the prediction market scenario, members of the population reveal information about their beliefs only through the bets that they choose to make, and it is up to the mechanism to entice them to reveal their true beliefs [62, 90]. One can imagine other variants of the problems described here in which the algorithm could have access to more or less information about the population, or even active learning variants [41, 9] in which the algorithm is endowed with the ability to query specific individuals about their preferences or beliefs.

1.2 Overview of This Dissertation

This dissertation proposes new theoretical models and algorithms designed to address some of the challenging problems introduced above. We now describe each of these problems in more detail and summarize the main technical results contained in this document.

1.2.1 Learning from Like-Minded Users

Most real-life collaborative filtering systems, like those used by Amazon and Netflix, rely on complicated combinations of ad hoc techniques with little theoretical justification [13, 14]. While there has been a surge of theoretical work on the use of low-rank matrix completion techniques for collaborative filtering [6, 117, 118], very little thought has been given to alternate theoretical frameworks.

In Chapter 2, we approach this problem from a different angle. Building on the basic foundations of learning theory [76, 125] , we develop a full PAC-style theory of learning from multiple sources of similar data (in this case, website or movie ratings from multiple similar users). Our results illustrate the fundamental trade-offs that arise when combining data from a set of users to learn a personalized model for one particular user.

More specifically, given distinct samples from multiple data sources and estimates of the dissimilarities between these sources, we provide a general theory of which samples should be used to learn models for each source, establishing a set of error bounds that clearly express a trade-off between three quantities: the sample size used, a weighted average of the disparities of the sources whose data is used, and a model complexity term. These bounds apply in a wide range of learning paradigms, including classification, regression, and in some cases, density estimation. In fact, the theory can be applied to any learning setting in which the loss function obeys an "approximate" triangle inequality and the hypothesis class under consideration obeys uniform convergence of empirical estimates of loss to their expectations.

We also briefly turn to more recent work on the related problem of domain adaptation with multiple sources. The key distinction between this setting and those mentioned above is that here the *underlying distribution* over data points is different for each source, while the labeling functions are assumed to be similar. As an example of a situation in which we might expect these assumptions to hold, suppose that we would like to build a personalized spam filter for each user of an email system. Here we might guess that any pair of users are likely to agree on which messages should be considered spam, while the distribution over email they receive could be quite different. We describe uniform convergence bounds in this setting for algorithms that minimize a convex combination of empirical error on each source of data.

1.2.2 Learning from Collective Behavior

Sociologists, economists, and researchers in a variety of other fields have spent decades studying the collective behavior of large populations in countless domains. The result is an impressive literature on models of collective behavior for phenomena as diverse as herding behavior in financial markets [131], diffusion of government policies such as anti-smoking laws or state lottery adoption [114, 119], the spread of new agricultural or medical practices [110, 36] or Hollywood trends [43], and the contagion properties of obesity [33]. More recently, collective behavior has attracted the attention of computer scientists, leading to work on viral marketing [108, 94], information propagation on blogs [96, 60], the transmission of infectious diseases or computer viruses [46, 18], and the prevention of water contamination [95]. The mathematical details of these models vary dramatically, but they all share the underlying assumption that each agent's current behavior is entirely or largely determined by the recent behavior of the other agents. The population evolves over time according to its own internal dynamics.

Inspired by this exciting line of work, in Chapter 3, we introduce and describe a new computational theory of learning from collective behavior. In our model, each agent i acts according to a fixed but unknown strategy c_i drawn from a known class $\mathcal{C}$. A strategy probabilistically maps the current state of the population (or the state of the agent's local neighborhood, if a network structure is defined over the population) to the next state or action for that agent, and each agent's strategy may be different. The goal of the learning algorithm is to accurately model and predict the future behavior of a large population after observing their interactions during a training phase of polynomial length.

As an example of the type of interaction we have in mind, consider a population of students who must each decide which local bar to patronize each night. Each student is faced with the task of balancing his desire to frequent the current hot spot with his own intrinsic preferences based on decor or price. Similarly, an American citizen voting in a presidential election might alter her anticipated voting decision over time in response to primary or polling news, balancing her intrinsic preferences over the candidates with a desire to avoid wasting a vote on an "unelectable" candidate. We consider models in which agents integrate these sometimes opposing forces when deciding how to behave at each moment, and study the problem of how a learning algorithm watching the collective behavior of such a population might produce an accurate model of their future behavior.

We start by defining a formal model for efficient learning in such settings, and go on to develop general theory for this model. Our main result is a polynomial-time reduction of learning from collective behavior to more traditional i.i.d. learning. We then define specific classes of agent strategies, including "crowd affinity" strategies (in which agents balance personal preferences with a desire to be like the crowd) and complementary "crowd aversion" strategies (in which agents prefer to stand out from the crowd), and provide provably efficient algorithms for learning from collective behavior for these classes. We also discuss some natural variants of the model, and describe how to extend our results to these alternative settings.

1.2.3 Aggregating Opinions Via Expert Advice

Suppose that every evening, we would like to predict whether or not it is going to rain the following day. We might base our prediction on the opinions of friends or coworkers, reports from meteorologists, advice from newscasters, and so on. Each of these "experts" is sometimes right and sometimes wrong, and there's no way of knowing a priori whose predictions will be best. In such a setting, a natural goal that we might consider is to be able to combine the predictions of our sources in such a way that our own predictions won't be too much worse than those of the source who predicted best in retrospect.

As described above, the literature on no-regret learning shows that it is possible to do just that. In particular, on any sequence of T trials in which the predictions of a set of N experts are observed, it is possible to guarantee a cumulative reward that is within $O(\sqrt{T \log N})$ of the reward of the best single expert in hindsight using a simple dynamically weighted prediction.[1] Somewhat strikingly, this result holds even in a fully adversarial setting in which no distributional assumptions are made about the performance of the experts.

However, despite the impressiveness of these results, there are many situations in which competing with the best individual in the population is not good enough. Consider the following simple example, in which there are only two experts. The rewards for expert 1 alternate $1, 0, 1, 0, \cdots$, while the rewards for expert 2 alternate $0, 1, 0, 1, \cdots$. Due to their aggressive updates, standard regret minimization algorithms (including Exponential Weights [98, 55], Follow the Perturbed

[1] In this setting, the "rewards" can be thought of as scores based on how accurate each expert's predictions are. For example, the reward might be 1 for each correct prediction and 0 for each incorrect prediction.

Leader [72], and Prod [26]) yield a cumulative reward of $T/2 - \Theta(\sqrt{T})$, meeting their guarantee of $O(\sqrt{T})$ regret with respect to the best expert. However, this performance leaves something to be desired. In this example, where both experts have similar performance, all of the algorithms above end up suffering $\Omega(\sqrt{T})$ regret to the *worst* expert as well.

In Chapter 4, we examine no-regret learning in a *bicriteria* setting. We analyze not only the standard notion of regret to the best expert, but also the regret to the average of all experts, the regret to any fixed mixture of experts, and the regret to the worst expert. We show that *any* algorithm that achieves only $O(\sqrt{T})$ cumulative regret to the best expert on a sequence of T trials must, in the worst case, suffer regret $\Omega(\sqrt{T})$ to the average, and that for a wide class of update rules that includes many existing no-regret algorithms (such as Exponential Weights and Follow the Perturbed Leader), the product of the regret to the best and the regret to the average is, in the worst case, $\Omega(T)$. We then describe and analyze two alternate new algorithms that both achieve cumulative regret only $O(\sqrt{T}\log T)$ to the best expert and have only *constant* regret to any given fixed distribution over experts. The key to achieving such guarantees is to allow the aggressiveness of the algorithm to change over time, updating more aggressively only when it becomes clear that one expert is dominating the rest.

These results demonstrate the inherent tension between aggressively following the current best expert (the "needle in a haystack" approach) and not changing weight too quickly when there are small fluctuations in expert performance. We show that existing algorithms frequently manage this trade-off poorly, while our new algorithms enjoy optimal bicriteria performance guarantees.

1.2.4 Aggregating Opinions Via Prediction Markets

As described above, prediction markets are betting markets designed to aggregate beliefs about the outcome of a future event into a single accurate prediction. Most prediction markets operate over relatively small outcome spaces. A typical horse race market might allow bettors to choose one of n horses as the expected winner, ignoring the fact that $n!$ distinct outcomes are possible if we choose to consider all possible permutations of horses in the race. One could argue that such simplifications are necessary. It is difficult for humans to reason about large outcome spaces, and computationally demanding to store and update an exponential number of prices. However, restricting the betting language in other ways (for example, allowing only bets of the form "horse A

will either come in first place or third place") can simplify the reasoning process for bettors while simultaneously making price computations tractable in certain types of markets [29].

In Chapter 5, we investigate the computational complexity of market maker pricing algorithms for these "combinatorial" prediction markets. We restrict our attention to the popular logarithmic market scoring rule market maker (LMSR) introduced by Hanson [62, 63]. Our goal is to implicitly maintain correct LMSR prices across an exponentially large outcome space. We examine both permutation combinatorics, where outcomes are permutations of objects (as is the case in a horse race or an election), and Boolean combinatorics, where outcomes are combinations of binary events, and show that even with severely limited languages, LMSR pricing is $\#\mathsf{P}$-hard. These results contrast with the results of Chen et al. [29], who show that solving the auctioneer's matching problem can be done in polynomial time for one of the same languages.

We go on to demonstrate and study the previously unexplored connection between LMSR prices and the weights maintained by algorithms for learning from expert advice. We propose an approximation technique for pricing permutation markets which takes advantage of known results for online permutation learning [67]. We believe that this striking connection between two disjoint fields may be of independent interest, opening up new directions of future research; see Chapter 6.

1.3 Bibliographic Notes

The model and analysis of learning from like-minded users in Chapter 2 are based primarily on joint work with Koby Crammer and Michael Kearns [40]. The extensions discussed in Section 2.8 grew out of work with John Blitzer, Koby Crammer, Alex Kulesza, and Fernando Pereira [20]. The model and analysis of learning from collective behavior in Chapter 3 are based on joint work with Michael Kearns [77]. The new perspective and analysis of learning from expert advice in Chapter 4 are based on joint work with Eyal Even-Dar, Michael Kearns, and Yishay Mansour [52]. Finally, the work on pricing problems in combinatorial prediction markets and the connection to no-regret learning presented in Chapter 5 are based on joint work with Yiling Chen, Lance Fortnow, Nicolas Lambert, and David Pennock [31]. All results, figures, and text that have been published elsewhere are included with the permission of all authors.

Chapter 2

Learning from Like-Minded Users

Over the past decade, the increasing popularity of e-commerce and online shopping sites has led to the launch of countless product recommendation systems. These systems provide web users with personalized suggestions for books, movies, music, and more. Netflix offers each of its users individual movie recommendations based on the user's own preferences and the preferences of other users with similar taste. Amazon offers a variety of personalized product recommendations to each user based on their previous shopping habits and the set of items that other similar shoppers have viewed or purchased. Such social recommendation or *collaborative filtering* systems work reasonably well in practice, but are frequently based on conglomerations of ad hoc techniques [13, 14]. While there have been some recent theoretical advancements on collaborative filtering, these have largely focused on techniques for low-rank matrix completion [6, 117]. There remains little foundational understanding of why the complicated systems used in practice work as well as they do, or how to make them better.

The work presented in this chapter can be seen as a step towards gaining this understanding. Here we build upon the foundations of learning theory and examine the fundamental trade-offs that arise when combining data from a set of users to learn a personalized model for a single user. The specific problem we analyze is the somewhat more general problem of learning accurate models from multiple sources of "nearby" data. In particular, given distinct samples from multiple data sources and estimates of the dissimilarities between these sources, we provide a general theory of which samples should be used to learn models for each source. This theory is applicable in a broad decision-theoretic learning framework, and yields general results for classification and regression.

Most of the work described in this chapter was done in collaboration with Koby Crammer and Michael Kearns [40]. The final section describes joint work with John Blitzer, Koby Crammer, Alex Kulesza, and Fernando Pereira [20].

2.1 Overview

Suppose that for each web user in a large population, we wish to learn a classifier to predict which sites that user is likely to find interesting. Assuming we have at least a small amount of labeled data for each user (as might be obtained either through direct feedback, or via indirect means such as click-throughs following a search), one approach would be to apply standard learning algorithms to each user's data in isolation. However, if there are natural and accessible measures of similarity between the interests of pairs of users (as might be obtained through their mutual labellings of common web sites), an appealing alternative is to *aggregate* the data of "similar" users when learning a classifier for each particular user. This alternative is intuitively subject to a trade-off between the increased sample size and how different the aggregated users are.

We treat this problem in some generality and provide a bound addressing the aforementioned trade-off. For the majority of this chapter, we consider a model in which there are K unknown data sources, with source i generating a distinct sample S_i of n_i observations. We assume we are given only the samples S_i, and a *disparity*[1] matrix D whose entry $D(i, j)$ bounds the difference between source i and source j. Given these inputs, we wish to decide which subset of the samples S_j will result in the best model for each source i. Our framework includes settings in which the sources produce data for classification, regression, and in some special cases, density estimation (and more generally any additive-loss learning problem obeying certain conditions).

Our main result is a general theorem establishing a bound on the expected loss incurred by using all data sources within a given disparity of the target. Optimization of this bound then yields a recommended subset of the data to be used in learning a model of each source. Our bound clearly expresses a trade-off between three quantities: the sample size used (which increases as we include data from more distant models), a weighted average of the disparities of the sources whose data is used, and a model complexity term. It can be applied to any learning setting in which the

[1] We avoid using the term distance since our results include settings in which the underlying loss measures may not be formal distances.

underlying loss function obeys an *approximate* triangle inequality and the class of hypothesis models under consideration obeys uniform convergence of empirical estimates of loss to expectations. For classification problems, the standard triangle inequality holds. For regression we prove a 2-approximation to the triangle inequality. Uniform convergence bounds for the settings we consider may be obtained via standard data-independent model complexity measures such as VC dimension and pseudo-dimension, or via data-dependent measures such as Rademacher complexity.

The final section of this chapter touches on more recent work on the related problem of domain adaptation with multiple sources. Suppose that our goal is no longer to produce personalized product recommendations, where individual tastes and preferences are crucial, but instead to build a personalized spam filter for each user of an email system. We might assume that any pair of users are likely to agree on which messages should be considered spam. However, the underlying *distribution* over email received by two users could be quite different. We briefly present uniform convergence bounds in this setting for algorithms that minimize a convex combination of empirical error on each source.

The primary framework examined in this chapter can be viewed as a model for rudimentary collaborative filtering. While there have been some theoretical developments in collaborative filtering in recent years, almost all of them deal with the problem of low-rank matrix completion [6, 117, 118], in which each known preference rating is entered into a matrix, with rows representing users and columns representing movies, and missing entries are then approximated in such a way that the completed matrix has low rank. This work makes the implicit assumptions that preferences can be decomposed into a small number of unknown factors and that it is not necessary to explicitly define features in order to find this decomposition. While there may be settings in which these assumptions hold, incorporating available feature data can be crucial when the amount of training data is limited; as an extreme example, there is no hope of estimating a user's rating for a website no other user in the system has rated before unless additional feature data is considered. There is little theoretical work on alternate approaches to this problem.

This work can also be viewed as a specific type of multi-task learning or transfer learning [12, 15, 101]. Wu and Dietterich [133] studied a related problem experimentally in the context of SVMs. In earlier work [39], we examined the considerably more limited problem of learning a model when all data sources are corrupted versions of a *single, fixed* source, for instance when

each data source provides noisy samples of a fixed binary function, but with varying levels of noise. On the contrary, here the labels on each source may be entirely unrelated to those on other source except as constrained by the bounds on disparities, requiring us to develop new and more general techniques.

Chapter Outline: In the next section, we introduce a decision-theoretic framework for probabilistic learning that includes classification, regression, density estimation, and many other settings as special cases, and then give our multiple source generalization of this model. In Section 2.3 we provide our main result, which is a general bound on the expected loss incurred by using all data within a given disparity of a target. Section 2.4 discusses the most simple application of this bound to binary classification using VC theory. In Section 2.5, we give applications of our general theory to classification and regression using Rademacher complexity, and show more generally how the theory can be applied for any Lipschitz loss function. In Section 2.6 we discuss the important detail of how to empirically estimate the disparity matrix from data. In Section 2.7, we illustrate the theory through synthetic simulations. In Section 2.8, we mention more recent work on domain adaptation from multiple sources. Finally, in Section 2.9, we mention some related open directions of research.

2.2 The Learning Model

Before detailing our multiple-source learning model, we first introduce a standard decision-theoretic learning framework in which our goal is to find a model minimizing a generalized notion of empirical loss [65]. Let the *hypothesis class* $\mathcal{H}$ be a set of models (which might be classifiers, real-valued functions, densities, etc.), and let f be the *target model*, which may or may not lie in the class $\mathcal{H}$. Let z be a (generalized) data point or observation. For instance, in noise-free classification and regression, z consists of a pair $\langle x, y \rangle$ where $y = f(x)$. We assume that the target model f induces some underlying distribution P_f over observations z. In the case of classification or regression, P_f is induced by drawing the inputs x according to some underlying distribution P, and then setting $y = f(x)$ (possibly corrupted by noise).

Each setting we consider has an associated *loss function* $\mathcal{L}(h, z)$. For example, in classification we typically consider the 0/1 loss: $\mathcal{L}(h, \langle x, y \rangle) = 0$ if $h(x) = y$, and 1 otherwise. In regression we

might consider the squared loss function $\mathcal{L}(h, \langle x, y\rangle) = (y - h(x))^2$. In each case, we are interested in the expected loss of a model h_2 on target h_1, $e(h_1, h_2) = \mathrm{E}_{z\sim P_{h_1}}[\mathcal{L}(h_2, z)]$. Expected loss is not necessarily symmetric.

In our multiple source model, we are presented with K distinct mutually independent samples or *sources* of data $S_1, ..., S_K$, and a symmetric $K \times K$ matrix D. Each source S_i contains n_i observations that are generated from a fixed and unknown model f_i, and D satisfies $\max(e(f_i, f_j), e(f_j, f_i)) \leq D(i, j)$. When D is unknown, it often can be estimated from a small amount of data; see Section 2.6 for more details. Our goal is to decide which sources S_j to use in order to learn the best approximation (in terms of expected loss) to each f_i.

While we are interested in accomplishing this goal for each f_i, it suffices and is convenient to examine the problem from the perspective of a fixed f_i. Thus without loss of generality let us suppose that we are given sources $S_1, ..., S_K$ of size $n_1, \ldots, n_K$ from models $f_1, \ldots, f_K$ such that $\epsilon_1 \equiv D(1, 1) \leq \epsilon_2 \equiv D(1, 2) \leq \cdots \leq \epsilon_K \equiv D(1, K)$, and our goal is to learn f_1. Here we have simply taken the problem in the preceding paragraph, focused on the problem for f_1, and reordered the other models according to our estimations or their proximity to f_1. To highlight the distinguished role of the target f_1 we shall denote it f. We denote the observations in S_j by $z_1^j, \ldots, z_{n_j}^j$. We analyze, for every $k \leq K$, the error of the hypothesis $\hat{h}_k$ minimizing the empirical loss $\hat{e}_k(h)$ on the first k sources $S_1, \ldots, S_k$, that is

$$\hat{h}_k = \operatorname*{argmin}_{h\in\mathcal{H}} \hat{e}_k(h) = \operatorname*{argmin}_{h\in\mathcal{H}} \frac{1}{n_{1:k}} \sum_{j=1}^{k} \sum_{i=1}^{n_j} \mathcal{L}(h, z_i^j) \,,$$

where we define the shorthand $n_{1:k} = n_1 + \cdots + n_k$. We also denote the expected error of function h with respect to the first k sources of data as

$$e_k(h) = \mathrm{E}\left[\hat{e}_k(h)\right] = \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) e(f_i, h).$$

2.3 General Theory for the Multiple Source Problem

In this section we provide the first of our main results: a general bound on the expected loss of the model minimizing the empirical loss on the nearest k sources. Optimization of this bound leads to

a recommended set of sources to incorporate when learning $f = f_1$. The key ingredients needed to apply this bound are an approximate triangle inequality and a uniform convergence bound, which we define below. In the subsequent sections we demonstrate that these ingredients can indeed be provided for a variety of natural learning problems.

Definition 1 *For $\alpha \geq 1$, we say that the α-**triangle inequality** holds for a class of models $\mathcal{F}$ and expected loss function e if for all $h_1, h_2, h_3 \in \mathcal{F}$ we have*

$$e(h_1, h_2) \leq \alpha(e(h_1, h_3) + e(h_3, h_2)).$$

The parameter $\alpha \geq 1$ is a constant that depends on $\mathcal{F}$ and e.

The choice $\alpha = 1$ yields the standard triangle inequality. We note that the restriction to models in the class $\mathcal{F}$ may in some cases be quite weak — for instance, when $\mathcal{F}$ is all possible classifiers or real-valued functions with bounded range — or stronger, as in densities from the exponential family. Our results require only that the unknown *source* models $f_1, \ldots, f_K$ lie in $\mathcal{F}$, even when our *hypothesis* models are chosen from some possibly much more restricted class $\mathcal{H} \subseteq \mathcal{F}$. For now we simply leave $\mathcal{F}$ as a parameter of the definition.

Definition 2 *A **uniform convergence bound** for a hypothesis space $\mathcal{H}$ and loss function $\mathcal{L}$ is a bound that states that for any $0 < \delta < 1$, with probability at least $1 - \delta$ for any $h \in \mathcal{H}$*

$$|\hat{e}(h) - e(h)| \leq \beta(n, \delta) \,,$$

where $\hat{e}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(h, z_i)$ for n observations $z_1, \ldots, z_n$ generated independently according to distributions $P_1, \ldots P_n$, and $e(h) = \mathrm{E}\left[\hat{e}(h)\right]$ where the expectation is taken with respect to $z_1, \ldots, z_n$. Here β is a function of the number of observations n and the confidence δ, and depends on $\mathcal{H}$ and $\mathcal{L}$.

This definition simply asserts that for every model in $\mathcal{H}$, its empirical loss on a sample of size n and the expectation of this loss are "close" when $\beta(n, \delta)$ is small. In general the function β incorporates standard measures of the complexity of $\mathcal{H}$, and is a decreasing function of the sample size n, as in the classical $O(\sqrt{d/n})$ bounds of VC theory. Our bounds are derived from the rich

literature on uniform convergence. The only twist to our setting is the fact that the observations are no longer necessarily identically distributed, since they are generated from multiple sources. However, generalizing the standard uniform convergence results to this setting is mostly straightforward as we show in the upcoming sections on applications of the bound.

We are now ready to present our general bound.

Theorem 1 *Let e be the expected loss function for loss $\mathcal{L}$, and let $\mathcal{F}$ be a class of models for which the α-triangle inequality holds with respect to e. Let $\mathcal{H} \subseteq \mathcal{F}$ be a class of hypothesis models for which there is a uniform convergence bound β for $\mathcal{L}$. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\epsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as above. For any δ such that $0 < \delta < 1$, with probability at least $1 - \delta$, for any $k \in \{1, \ldots, K\}$*

$$e(f, \hat{h}_k) \leq \alpha^2 \min_{h \in \mathcal{H}} \{e(f, h)\} + (\alpha + \alpha^2) \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \epsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) \, .$$

Before providing the proof, let us examine the bound of Theorem 1, which expresses a natural and intuitive trade-off. The first term in the bound is simply the *approximation error*, the residual loss that we incur by limiting our hypothesis model to fall in the restricted class $\mathcal{H}$. The second term is a weighted sum of the disparities of the $k \leq K$ models whose data is used with respect to the target model $f = f_1$. We expect this term to *increase* as we increase k to include more distant sources. The final term is determined by the uniform convergence bound. We expect this term to *decrease* with added sources due to the increased sample size. All three terms are influenced by the strength of the approximate triangle inequality that we have, as quantified by α.

The bound given in Theorem 1 can be loose, but provides an upper bound necessary for optimization and suggests a natural choice for the number of sources k^* to use to estimate the target:

$$k^* = \operatorname*{argmin}_k \left((\alpha + \alpha^2) \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \epsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) \right) .$$

Theorem 1 and this optimization make the implicit assumption that the best subset of sources to use is a prefix of the sources — that is, that we should not "skip" a nearby source in favor of more distant ones. This assumption is true for typical data-independent uniform convergence bounds, and is true on average for data-dependent bounds, where we expect uniform convergence bounds

to improve with increased sample size. We now give the proof of Theorem 1.

Proof of Theorem 1: By Definition 1, for any $h \in \mathcal{H}$, any $k \in \{1, \ldots K\}$, and any $i \in \{1, \ldots, k\}$,

$$\left(\frac{n_i}{n_{1:k}}\right) e(f,h) \leq \left(\frac{n_i}{n_{1:k}}\right) \left(\alpha e(f,f_i) + \alpha e(f_i,h)\right) \ .$$

Summing over all $i \in \{1, \ldots, k\}$, we find

$$\begin{aligned} e(f,h) &\leq \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \left(\alpha e(f,f_i) + \alpha e(f_i,h)\right) \\ &= \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) e(f,f_i) + \ \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) e(f_i,h) \leq \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \epsilon_i + \alpha e_k(h) \ . \end{aligned}$$

In the first line above we have used the α-triangle inequality to deliberately introduce a weighted summation involving the f_i. In the second line, we have broken up the summation using the fact that $e(f,f_i) \leq \epsilon_i$ and the definition of $e_k(h)$. Notice that the first summation is a weighted average of the expected loss of each f_i, while the second summation is the expected loss of h on the data. Using the uniform convergence bound, we may assert that with high probability $e_k(h) \leq \hat{e}_k(h) + \beta(n_{1:k}, \delta/2K)$, and with high probability

$$\hat{e}_k(\hat{h}_k) = \min_{h \in \mathcal{H}} \{\hat{e}_k(h)\} \leq \min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) e(f_i,h) + \beta(n_{1:k}, \delta/2K) \right\} \ .$$

Putting these pieces together, we find that with high probability

$$\begin{aligned} e(f,\hat{h}_k) &\leq \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \epsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) + \alpha \min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) e(f_i,h) \right\} \\ &\leq \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \epsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) \\ &\quad + \alpha \min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \alpha e(f_i,f) + \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \alpha e(f,h) \right\} \\ &= (\alpha + \alpha^2) \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}}\right) \epsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) + \alpha^2 \min_{h \in \mathcal{H}} \{e(f,h)\} \ . \end{aligned}$$

■

2.4 Simple Application to Binary Classification

We demonstrate the applicability of the general theory given by Theorem 1 to several standard learning settings. As a warm-up, we begin with the most straightforward application, classification using VC bounds.

In (noise-free) binary classification, we assume that our target model is a fixed, unknown and arbitrary function f from some input set $\mathcal{X}$ to $\{0,1\}$, and that there is a fixed and unknown distribution P on the $\mathcal{X}$. Note that the distribution P over input does not depend on the target function f. The observations are of the form $z = \langle x, y\rangle$ where $y \in \{0,1\}$. The loss function $\mathcal{L}(h, \langle x, y\rangle)$ is defined as 0 if $y = h(x)$ and 1 otherwise, and the corresponding expected loss is $e(h_1, h_2) = \mathrm{E}_{\langle x,y\rangle \sim P_{h_1}}\left[\mathcal{L}(h_2, \langle x, y\rangle)\right] = \Pr_{x\sim P}\left[h_1(x) \neq h_2(x)\right]$.

For 0/1 loss it is well-known and easy to see that the (standard) 1-triangle inequality holds. Classical VC theory [125] provides us with uniform convergence as follows.

Lemma 1 *Let $\mathcal{H} : \mathcal{X} \to \{0,1\}$ be a class of functions with VC dimension d, and let $\mathcal{L}(h, \langle x, y\rangle) = |y - h(x)|$ be the 0/1-loss. The following function β is a uniform convergence bound for $\mathcal{H}$ and $\mathcal{L}$ when $n \geq d/2$:*

$$\beta(n, \delta) = \sqrt{\frac{8(d\ln(2\mathrm{e}n/d) + \ln(4/\delta))}{n}}\,.$$

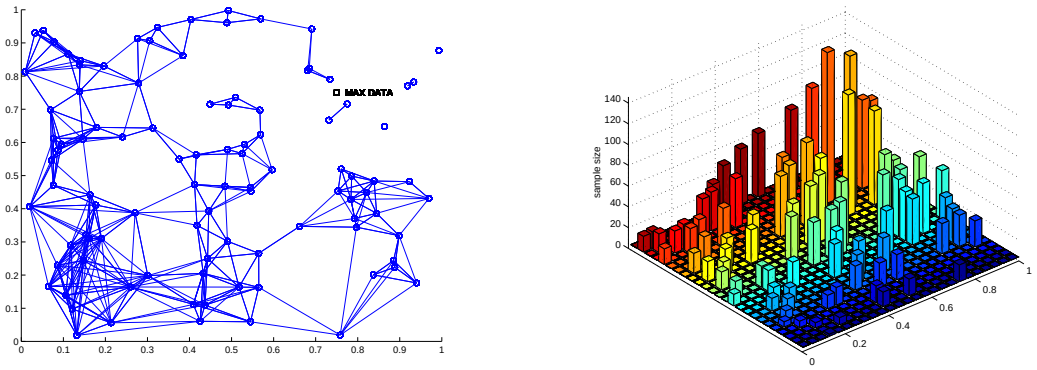
The proof is analogous to the standard proof of uniform convergence using the VC Dimension (see, for example, Chapters 2–4 of Anthony and Bartlett [4]), requiring only minor modifications to the symmetrization argument to handle the fact that the samples need not be uniformly distributed.

With Lemma 1 in place, the conditions of Theorem 1 are easily satisfied, yielding the following result.

Theorem 2 *Let $\mathcal{F}$ be the set of all functions from an input set $\mathcal{X}$ into $\{0,1\}$ and let d be the VC dimension of $\mathcal{H} \subseteq \mathcal{F}$. Let e be the expected 0/1 loss. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\epsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as above in the multi-source learning model, and assume that $n_1 \geq d/2$. For any δ such that $0 < \delta < 1$, with probability at least $1-\delta$, for any $k \in \{1, \ldots, K\}$*

$$e(f, \hat{h}_k) \leq \min_{h\in\mathcal{H}}\{e(f,h)\} + 2\sum_{i=1}^{k}\left(\frac{n_i}{n_{1:k}}\right)\epsilon_i + \sqrt{\frac{32\left(d\ln\left(2\mathrm{e}n_{1:k}/d\right) + \ln\left(8K/\delta\right)\right)}{n_{1:k}}}\,.$$

Figure 2.1 Visual illustration of Theorem 2. See the text for details.



In Figure 2.1 we provide a visual illustration of the behavior of Theorem 2 applied to a simple classification problem. In this problem there are $K = 100$ classifiers, each classifier f_i for $i = 1 \ldots 100$ is defined by 2 parameters represented by a point in the unit square, such that the expected disagreement rate between two such classifiers is proportional the L_1 distance between their parameters.[2] We chose the 100 parameter vectors f_i uniformly at random from the unit square (the circles in the left panel). To generate varying source sizes, we let n_i decrease with the distance of f_i from a chosen "central" point at $(0.75, 0.75)$ (marked "MAX DATA" in the left panel); the resulting source sizes for each model are shown in the bar plot in the right panel, where the origin $(0, 0)$ is in the near corner, $(1, 1)$ in the far corner, and the source sizes clearly peak near $(0.75, 0.75)$. For every function f_i we used Theorem 2 to find the best sources j to be used to estimate its parameters. The undirected graph on the left includes an edge between f_i and f_j if and only if the data from f_j is used to learn f_i and/or the converse.

The graph simultaneously displays the geometry implicit in Theorem 2 as well as its adaptivity to local circumstances. Near the central point, the graph is sparse and the edges quite short, corresponding to the fact that for such models we have enough direct data (represented with high bars in the right panel) that it is not advantageous to include data from distant models. Far from the central point the graph becomes dense and the edges long, as we are required to aggregate a larger neighborhood to learn the optimal model. In addition, decisions are affected locally by how many

[2] It is easy to create simple input distributions and classifiers that generate exactly this geometry. For example, let the input x be a pair $x = (p, b)$ where $p \in [0, 1], b \in \{0, 1\}$ and let the hypothesis class consist of functions defined as pairs of thresholds $f = (t_1, t_2)$ where $f(x) = 1$ if and only if ($p > t_1$ and $b = 0$) or ($p > t_2$ and $b = 1$). The distribution of $x = (p, b)$ is a product of a uniform distribution for p and a fair coin for b.

models are "nearby" a given model, when there are many close functions f_j to a given f_i there is no need to use "far" models, but when the neighborhood of a function is not populated with many examples, there is a need for data from models far-away.

2.5 Bounds Using Rademacher Complexity

Given the recent interest in tighter, potentially data-dependent convergence bounds such as maximum margin bounds, PAC-Bayes, and others, it is natural to ask how our multi-source theory can exploit them. We examine one specific case here using Rademacher complexity [10, 11, 87, 88]; analogs can be derived in a similar manner for other complexity measures. We start by deriving bounds for settings in which generic Lipschitz loss functions are used, and then discuss specific applications to classification and to regression with squared loss.

2.5.1 Rademacher Complexity and General Lipschitz-Loss Bounds

If $\mathcal{H}$ is a class of functions mapping from a set $\mathcal{X}$ to $\mathbb{R}$, the *empirical Rademacher complexity* of $\mathcal{H}$ on a fixed set of observations $x_1, \ldots, x_n$ is defined as

$$\hat{R}_n(\mathcal{H}) = \mathrm{E}\left[\sup_{h \in \mathcal{H}} \left| \frac{2}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right| \right] ,$$

where the expectation is taken with respect to independent uniform $\{\pm 1\}$-valued random variables $\sigma_1, \ldots, \sigma_n$. The *Rademacher complexity* for n observations can then be defined as $R_n(\mathcal{H}) = \mathrm{E}\left[\hat{R}_n(\mathcal{H})\right]$ where the expectation is with respect to observations $x_1, \ldots, x_n$. At a high level, the Rademacher complexity quantifies the extent to which a function in the class $\mathcal{H}$ can be correlated with a sequence of noise of length n, and thus how much overfitting is likely to take place when selecting models from this class.

In the standard setting, $x_1, \ldots, x_n$ are assumed to be i.i.d. (independently and identically distributed), drawn from a single fixed distribution. In our setting, these observations are still independent, but not necessarily identically distributed. We show that the standard uniform convergence results still hold in this slightly modified setting.

Consider any setting in which each generalized data point $z = \langle x, y \rangle$ for some $x \in \mathcal{X}$ and

$y \in \mathcal{Y}$ with $y = f(x)$. A *cost function* for the loss $\mathcal{L}$ is any function ϕ such that $\mathcal{L}(h, \langle x, y \rangle) = \phi(y, h(x))$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $h \in \mathcal{H}$. We consider cost functions ϕ that are Lipschitz in the second parameter. Define $\phi'(y, a) = \phi(y, a) - \phi(y, 0)$. If ϕ is Lipschitz in the second parameter with constant L then ϕ' is also Lipschitz in the second parameter with the same constant L.

Lemma 3 below gives a uniform convergence bound for any loss function with a corresponding Lipschitz cost function. The proof of this lemma is in Appendix A2.1. It is analogous to the proof of Theorem 8 in Bartlett and Mendelson [10], which makes a similar claim in the i.i.d. setting, and uses the following lemma.

Lemma 2 (Bartlett and Mendelson [10]) *If $f : \mathbb{R} \to \mathbb{R}$ is Lipschitz with constant L and $f(0) = 0$, then $R_n(f \circ \mathcal{H}) \leq 2LR_n(\mathcal{H})$.*

Lemma 3 *Let $\mathcal{L}$ be a loss function bounded in $[0, 1]$, and ϕ a cost function such that $\mathcal{L}(f, \langle x, y \rangle) = \phi(y, f(x))$ where ϕ is Lipschitz in the second parameter with constant L. Let $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to $\mathcal{Y}$ and let $\{\langle x_i, y_i \rangle\}_{i=1}^n$ be sampled independently. For any n, for any $0 < \delta < 1$, with probability $1 - \delta$ over samples of length n, every $h \in \mathcal{H}$ satisfies*

$$\beta(n, \delta) = 2LR_n(\mathcal{H}) + \sqrt{\frac{2\ln(2/\delta)}{n}}\,.$$

2.5.2 Application to Classification Using Rademacher Complexity

Theorem 3 below follows from the application of Theorem 1 using the 1-triangle inequality and an application of Lemma 3 with

$$\phi(y, a) = \begin{cases} 1 & \text{if } ya \leq 0, \\ 1 - ya & \text{if } 0 < ya \leq 1, \\ 0 & \text{if } ya > 1. \end{cases}$$

The middle condition is necessary only to enforce that ϕ is Lipschitz with constant 1. If $\mathcal{L}$ is the 0/1 loss, then for all $x \in \mathcal{X}$, $y \in \{-1, 1\}$, and $h \in \mathcal{X} \to \{-1, 1\}$, we have that $ya \in \{0, 1\}$ and thus $\mathcal{L}(h, \langle x, y \rangle) = \phi(y, h(x))$, so Lemma 3 can be applied immediately.

Theorem 3 *Let $\mathcal{F}$ be a set of functions from an input set $\mathcal{X}$ into $\{-1,1\}$ and let $R_{n_{1:k}}(\mathcal{H})$ be the Rademacher complexity of $\mathcal{H} \subseteq \mathcal{F}$ on the first k sources of data. Let e be the expected 0/1 loss. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\epsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as in the multi-source learning model. For any δ such that $0 < \delta < 1$, with probability at least $1 - \delta$, for any $k \in \{1, \ldots, K\}$*

$$e(f, \hat{h}_k) \leq \min_{h \in \mathcal{H}} \{e(f, h)\} + 2 \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \epsilon_i + 2\sqrt{\frac{2\ln(4K/\delta)}{n_{1:k}}} + 4R_{n_{1:k}}(\mathcal{H}) \, .$$

Before moving on, let us briefly examine the behavior of this bound. Similarly to the VC-based bound given in Theorem 2, as k increases and more sources of data are combined, the second term grows while the third shrinks. The behavior of the final term $R_{n_{1:k}}(\mathcal{H})$, however, is less predictable and may grow or shrink as more sources of data are combined.

Note that for the special case of classification with 0/1 loss, it is possible to get tighter bounds with better dependence on $R_{n_{1:k}}$ by using a more careful analysis than the one in the proof of Lemma 3. Such bounds are given in an early version of this work [40]; we choose not to present these alternate bounds here to simplify presentation.

2.5.3 Regression

We now turn to (noise-free) regression with squared loss. Here our target model f is any function from an input class $\mathcal{X}$ into some bounded subset of $\mathbb{R}$. (Frequently we have $\mathcal{X} \subseteq \mathbb{R}^d$, but this is not required.) Our loss function is $\mathcal{L}(h, \langle x, y \rangle) = (y - h(x))^2$, and the expected loss is thus $e(h_1, h_2) = \mathrm{E}_{\langle x,y \rangle \sim P_{h_1}} \left[\mathcal{L}(h_2, \langle x, y \rangle) \right] = \mathrm{E}_{x \sim P} \left[(h_1(x) - h_2(x))^2 \right]$.

For regression it is known that the standard 1-triangle inequality does not hold. However, a 2-triangle inequality does hold and is stated in the following lemma.

Lemma 4 *Given any three functions $h_1, h_2, h_3 : \mathcal{X} \to \mathbb{R}$, a fixed and unknown distribution P on the inputs $\mathcal{X}$, and the expected loss $e(h_1, h_2) = \mathrm{E}_{x \sim P} \left[(h_1(x) - h_2(x))^2 \right]$,*

$$e(h_1, h_2) \leq 2 \left(e(h_1, h_3) + e(h_3, h_1) \right) .$$

Proof: By Jensen's inequality and the convexity of $x \mapsto x^2$, for any h_1, h_2, and h_3,

$$\begin{aligned} e(h_1, h_2) &= \mathrm{E}_{x \sim P}\left[(h_1(x) - h_2(x))^2\right] \\ &= \mathrm{E}_{x \sim P}\left[4\left(\frac{1}{2}(h_1(x) - h_3(x)) + \frac{1}{2}(h_3(x) - h_2(x))\right)^2\right] \\ &\leq \mathrm{E}_{x \sim P}\left[2(h_1(x) - h_3(x))^2 + 2(h_3(x) - h_2(x))^2\right] \\ &= 2\left(e(h_1, h_3) + e(h_3, h_1)\right) \ . \end{aligned}$$

■

We can derive a uniform convergence bound for squared loss using Rademacher complexity as long as the region $\mathcal{Y}$ is bounded. The proof is in Appendix A2.2.

Lemma 5 *Let $\mathcal{H} : \mathcal{X} \to [-B, B]$ be a class of functions, and let $\mathcal{L}(h, \langle x, y \rangle) = (y - h(x))^2$ be the squared loss. The following function β is a uniform convergence bound for $\mathcal{H}$ and $\mathcal{L}$:*

$$\beta(n, \delta) = 8BR_n(\mathcal{H}) + 4B^2\sqrt{\frac{2\ln(2/\delta)}{n}} \ .$$

Combining this with Lemma 4 and applying Theorem 1 yields the following.

Theorem 4 *Let $\mathcal{F}$ be the set of functions from $\mathcal{X}$ into $[-B, B]$, and $\mathcal{H} \subseteq \mathcal{F}$. Let e be the expected squared loss. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\epsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as in the multi-source learning model. For any δ such that $0 < \delta < 1$, with probability at least $1 - \delta$, for any $k \in \{1, \ldots, K\}$*

$$e(f, \hat{h}_k) \leq 4 \min_{h \in \mathcal{H}}\{e(f, h)\} + 6\sum_{i=1}^{k}\left(\frac{n_i}{n_{1:k}}\right)\epsilon_i + 32BR_{n_{1:k}}(\mathcal{H}) + 16B^2\sqrt{\frac{2\ln(4K/\delta)}{n_{1:k}}} \ .$$

2.5.4 Remarks on the Use of Data-Dependent Complexity Measures

The following lemma, which relates the true Rademacher complexity of a function class to its empirical Rademacher complexity, follows directly from Theorem 11 of Bartlett and Mendelson [10], the proof of which does not require samples to be identically distributed.

Lemma 6 *Let $\mathcal{H}$ be a class of functions mapping to $[-1, 1]$. For any integer n, for any $0 < \delta < 1$, with probability $1 - \delta$,*

$$\left| R_n(\mathcal{H}) - \hat{R}_n(\mathcal{H}) \right| \leq \sqrt{\frac{8 \ln(2/\delta)}{n}}.$$

This lemma immediately allows us to replace $R_n(\mathcal{H})$ with that data-dependent quantity $\hat{R}_n$ in any of the bounds above for only a small penalty.

While the use of data-dependent complexity measures can be expected to yield more accurate bounds and thus better decisions about the number k^* of sources to use, it is not without its costs in comparison to the more standard data-independent approaches. In particular, in principle the optimization of a data-dependent version of the bound given in Theorem 3 to choose k^* may actually involve running the learning algorithm on all possible prefixes of the sources, since we cannot know the data-dependent complexity term for each prefix without doing so. In contrast, the data-independent bounds can be computed and optimized for k^* without examining the data at all, and the learning performed only once on the first k^* sources. This is especially useful when the number of sources is very large, or when labels are not free but must be purchased at a price.

2.6 Estimating the Disparity Matrix

A potential drawback of the theory presented here is the need to estimate the disparity matrix D when it is unknown. However, it is often the case that this matrix can be estimated with many fewer labeled samples than are required for learning. In this section, we discuss how D can be estimated in the classification setting.

As before, consider the scenario in which each target function is a fixed, unknown and arbitrary function from some input set $\mathcal{X}$ to $\{-1, 1\}$, and assume that there is a fixed and unknown distribution P over $\mathcal{X}$. Suppose we are given m data points labeled by a pair of functions f_i and f_j, and let $\hat{e}(f_i, f_j)$ be the fraction of points on which the labels disagree. By Hoeffding's inequality (see Section A1.1), with probability $1 - \delta'$,

$$|\hat{e}(f_i, f_j) - e(f_i, f_j)| \leq \sqrt{\frac{\ln(2/\delta')}{2m}}\ .$$

Thus in order to approximate $e(f_i, f_j)$ with an error no more than ϵ, only $\ln(2/\delta')/(2\epsilon^2)$ commonly

labeled points are needed. Applying the union bound gives us the following lemma.

Lemma 7 *Let $\mathcal{F}$ be a set of functions from $\mathcal{X}$ into $\{-1, 1\}$, and suppose $f_1, \ldots, f_K \in \mathcal{F}$. Let e be the expected 0/1 loss. Suppose that for each pair $i, j \in \{1, \cdots, K\}$, there exist $m_{i,j} \geq m_0$ examples distributed according to P commonly labeled by f_i and f_j, where*

$$m_0 = \frac{2\ln(K) + \ln(2/\delta)}{2\epsilon^2}$$

for any δ such that $0 \leq \delta \leq 1$, and let $\hat{e}(f_i, f_j)$ be the fraction of commonly labeled examples on which f_i and f_j disagree. Then with probability $1 - \delta$, for all $i, j \in \{1, \cdots, K\}$, $|\hat{e}(f_i, f_j) - e(f_i, f_j)| \leq \epsilon$.

Using the lemma we set the upper bound on the mutual error $e(f_i, f_j)$ between the pair of function f_i and f_j to be $D_{i,j} = \hat{e}(f_i, f_j) + \epsilon$. With probability at least $1 - \delta$ these bound holds simultaneously for all i, j.

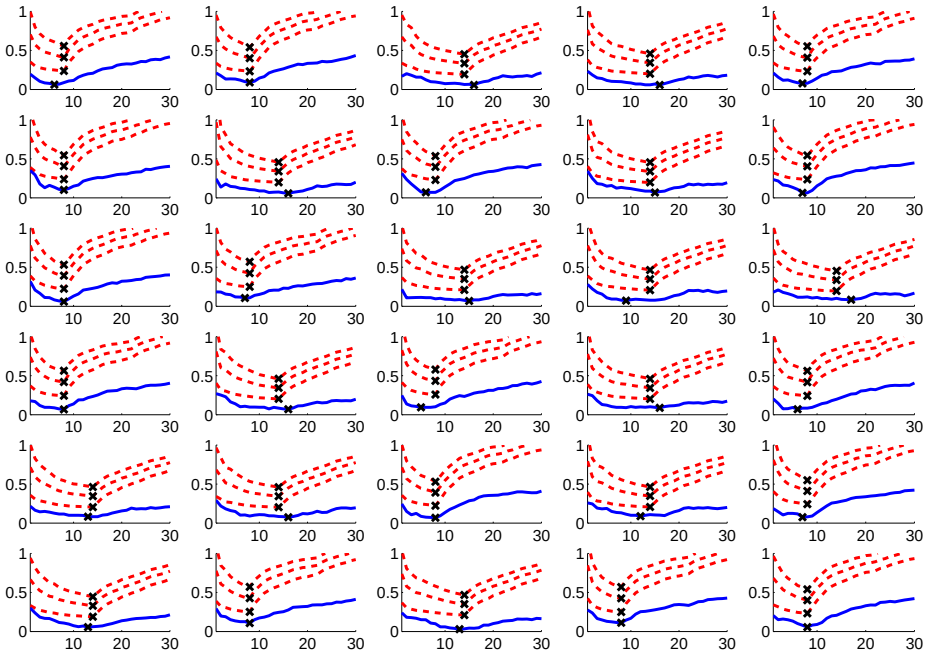
Note that in general, $\log(K)$ is significantly smaller than the dimension d of $\mathcal{H}$. Thus many fewer labeled examples are required to estimate the disparity matrix than to actually learn the best function in the class.

The assumption that there exist commonly labeled points for each pair of functions is natural in many settings. Consider, for example, the problem of predicting whether or not users will enjoy certain movies using ratings from other users. It is often the case that pairs of users have seen many of the same movies. These commonly rated movies can be used to determine how similar each pair of users are, while ratings of additional movies can be reserved to learn the prediction functions.

2.7 Synthetic Simulations

In this section, we illustrate the bounds of our main theorem through a simple synthetic simulation. Our hypothesis class $\mathcal{H}$ consists of all linear separators through the origin in 15 dimensions. The goal is to learn thirty classifiers from this class using only limited amounts of data. These data points are drawn uniformly at random from inside the 15-dimensional unit sphere. In this restricted setting, it is easy to calculate the disparity between two functions. Representing each function f

Figure 2.2 Simulation of the multiple source error bounds. See the text for details.



by a unit weight vector w such that $f(x) = \text{sign}(w \cdot x)$, the distance between functions w and w' is simply θ/π where $\theta = \arccos(w \cdot w')$ is the angle between w and w'.

In each simulation we ran, the linear classifiers were generated as follows. First, three base classifiers were generated by choosing weight vectors uniformly at random from the surface of the 15-dimensional sphere. Each of the thirty classifiers was then generated by randomly choosing one of the base classifiers, perturbing each coordinate of its weight vector, and renormalizing the perturbed weights.

The number of training samples available for each function was generated from a Poisson distribution with a mean of 8. Each data instance was then sampled from inside the 15-dimensional unit sphere via rejection sampling and labeled by the corresponding classifier, and 500 test samples for each function were generated in the same manner.

To predict the optimal set of training data sources to use for each model, we calculated an approximation of the multiple-source VC bound for classification. It is well known that the constants in the VC-based uniform convergence bounds are not tight. Thus for the purpose of illustrating how

these bounds might be used in practice, we have chosen to show approximations of our bounds with a variety of constants. In particular, we have chosen to approximate the bound with

$$2\sum_{i=1}^{k}\left(\frac{n_k}{n_{1:K}}\right)\epsilon_k + C\sqrt{\frac{(d\ln(2en_{1:K}/d)+\ln(8K/\delta))}{n_{1:K}}}$$

with $\delta = 0.001$ for different values of C. These approximations yield curves that are closer in shape and magnitude to the actual error than a curve generated using the precise, overly conservative constants of Theorem 2.

The set of plots shown in Figure 2.2 illustrates the results of a single multiple source simulation. (Results from repeated versions of this experiment and experiments with different source sizes were similar.) Each individual plot represents a particular target function. On the x axis is the number of data sources used in training. On the y axis is error. The solid curves (which appear blue in the full-color version of this document) show test error of a model trained using logistic regression. The dashed curves (red in the full-cover version) show our multiple source error bound with C set to $1/4$ in the lowest curve, $1/2$ in the middle curve, and $1/\sqrt{2}$ in the highest curve. The $\times$ on each curve marks the minimum value.

These plots clearly show the trade-off that exists. When too few sources are used, there is not enough data available to learn a 15-dimensional function. When too many sources are used, the labels on the training data often do not correspond to the labels that would have been assigned by the target function. The optimal amount of data lies somewhere in between.

Although the VC bounds remain loose even after constants have been dropped, the bounds tend to maintain the appropriate shape and thus predict the optimal set of sources quite well. In general, when C is set to small values, the predicted error values for small amounts of data (low k) tend to be quite accurate, while predicted values for larger amounts of data overestimate the true error. As C is set to larger values, the predictions become much larger in magnitude than the true error curves, but the shape of the prediction curves become more similar to the true error. In both cases, although the bounds are loose, they can still prove useful in determining the optimal set of sources to consider.

2.8 A Few Words on Domain Adaptation

We now very briefly turn to the related problem of domain adaptation with multiple sources, and describe a few more recent results. The key distinction between this setting and the one described above is that the *underlying distribution* over data points is different for each source, while the labeling functions are assumed to be similar. For example, suppose that we would like to combine data from many users in order to train a personalized spam filter for each user. While many users are likely to agree on whether or not a particular message should be labeled as spam, the distribution over a given user's email could be very different from the distribution over another's. Domain adaptation problems arise in a variety of other applications of machine learning too, and are especially common in face recognition systems [99], speech and acoustic modeling [92], and natural language processing [27, 21].

As before, we are presented with K distinct samples or sources of data $S_1, , \cdots, S_K$. Now each source S_i is associated with both an unknown labeling function f_i, and an unknown distribution $\mathcal{D}_i$ over input points. Source S_i contains n_i training instances distributed according to $\mathcal{D}_i$ and labeled by f_i. Our goal is to use these instances to train a model to perform well on a *target domain* $\langle \mathcal{D}_T, f_T \rangle$, which may or may not be the same domain as one of the sources.

For simplicity, we limit the discussion of domain adaptation to binary classification. For each source i, let $e_i(h)$ be the error of function h on source i, so $e_i(h) = \mathrm{E}_{x \sim \mathcal{D}_i}[|h(x) - f_i(x)|]$, and define e_T similarly for the target domain. Let $\hat{e}_i(h)$ be the empirical error of h on source i, i.e., the fraction of points in S_i for which h chooses the wrong label. We examine algorithms that minimize convex combinations of training error over the labeled examples from each source domain. Given a vector $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K)$ of nonnegative domain weights with $\sum_{i=1}^{K} \alpha_i = 1$, we define the empirical $\boldsymbol{\alpha}$-weighted error of function h as

$$\hat{e}_{\boldsymbol{\alpha}}(h) = \sum_{i=1}^{K} \alpha_i \hat{e}_i(h) = \sum_{i=1}^{K} \frac{\alpha_i}{n_i} \sum_{x \in S_i} |h(x) - f_i(x)| \, .$$

The true $\boldsymbol{\alpha}$-weighted error $e_{\boldsymbol{\alpha}}(h)$ is defined analogously. Finally, let $\mathcal{D}_{\boldsymbol{\alpha}}$ be a mixture of the K source distributions with mixing weights equal to the components of $\boldsymbol{\alpha}$.

We describe and contrast two alternative bounds on the error of the hypothesis that minimizes the $\boldsymbol{\alpha}$-weighted error. The first bound considers the quality and quantity of data available from

each source individually, ignoring the relationships between sources. On the contrary, the second bound (which also appears, with proof, in John Blitzer's dissertation [19]) depends directly on the distance between the target domain and a *weighted combination* of source domains. This allows us to achieve significantly tighter bounds when there exists a mixture of sources that approximates the target better than any single source.

In order to simplify the presentation of the trade-offs that arise, we state the bound in terms of VC dimension, but similar bounds could be obtained using alternate measure of complexity. Both measure the distance between domains using the $\mathcal{H}\Delta\mathcal{H}$-divergence,

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{h,h' \in \mathcal{H}} \left| \Pr_{x \sim \mathcal{D}} \left[h(x) \neq h'(x) \right] - \Pr_{x \sim \mathcal{D}'} \left[h(x) \neq h'(x) \right] \right| .$$

If the $\mathcal{H}\Delta\mathcal{H}$-divergence is small, pairs of functions that appear similar to each other on $\mathcal{D}$ also appear similar to each other on $\mathcal{D}'$. Additionally, this quantity can be efficiently approximated using a finite number of *unlabeled* samples of the distributions $\mathcal{D}$ and $\mathcal{D}'$ [16].

2.8.1 A Bound Using Pairwise Divergence

The first bound we present considers the pairwise $\mathcal{H}\Delta\mathcal{H}$-divergence between each source and the target, and illustrates the trade-off that exists between minimizing the average divergence of the training data from the target and weighting all points equally to encourage faster convergence. Somewhat surprisingly, the last term can be small even when there is not a single hypothesis that works well for all heavily weighted sources. The proof of this theorem is in Appendix A2.3.

Theorem 5 *Suppose we are given n_i labeled instances from source S_i for $i = 1 \dots K$. For a fixed vector of weights $\boldsymbol{\alpha}$, let $\hat{h} = \operatorname{argmin}_{h \in \mathcal{H}} \hat{e}_{\boldsymbol{\alpha}}(h)$, and let $h_T^* = \operatorname{argmin}_{h \in \mathcal{H}} e_T(h)$. Then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\begin{aligned} e_T(\hat{h}) \quad \leq \quad & e_T(h_T^*) + 2\sqrt{\sum_{i=1}^{K} \frac{\alpha_i^2}{2n_i} \left(d \log(2n_{1:K}) + \log(1/\delta) \right)} \\ & + \sum_{i=1}^{K} \alpha_i \left(2 \min_{h \in \mathcal{H}} \left\{ e_T(h) + e_i(h) \right\} + d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_T) \right) . \end{aligned}$$

In the special case in which the $\mathcal{H}\Delta\mathcal{H}$-divergence between each source and the target is 0 and all data instances are weighted equally, the bound in Theorem 5 becomes

$$e_T(\hat{h}) \leq e_T(h_T^*) + \sqrt{\frac{2\left(d\log(2n_{1:K}) + \log(1/\delta)\right)}{n_{1:K}}} + 2\sum_{i=1}^{K} \frac{n_i}{n_{1:K}} \min_{h\in\mathcal{H}} \{e_T(h) + e_i(h)\} \ .$$

Notice that this bound is nearly identical to Theorem 2. Aside from the constants in the complexity term, the only difference is that each ϵ_i term is replaced with $\min_{h\in\mathcal{H}} \{e_T(h) + e_i(h)\}$, which can be viewed as an alternate measure of label error between source S_i and the target, and is equivalent when the target function is a member of $\mathcal{H}$.

2.8.2 A Bound Using Combined Divergence

In the previous bound, the divergence between sources is measured pairwise, so it is not necessary to have a single hypothesis that is good for every source domain. However, this bound does not give us the flexibility to take advantage of domain structure when calculating unlabeled divergence. The alternate bound given in Theorem 6 allows us to effectively alter the source distribution by changing $\boldsymbol{\alpha}$. This has two consequences. First, we must now demand that there exists a hypothesis h^* which has low error on both the $\boldsymbol{\alpha}$-weighted convex combination of sources and the target domain. Second, we measure $\mathcal{H}\Delta\mathcal{H}$-divergence between the target and a mixture of sources, rather than between the target and each single source. The proof of this theorem is in Appendix A2.4.

Theorem 6 *Suppose we are given n_i labeled instances from source S_i for $i = 1 \ldots K$. For a fixed vector of weights $\boldsymbol{\alpha}$, let $\hat{h} = \operatorname{argmin}_{h\in\mathcal{H}} \hat{e}_{\boldsymbol{\alpha}}(h)$, and let $h_T^* = \operatorname{argmin}_{h\in\mathcal{H}} e_T(h)$. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\begin{aligned} e_T(\hat{h}) \quad \leq \quad & e_T(h_T^*) + 2\sqrt{\sum_{i=1}^{K} \frac{\alpha_i^2}{2n_i}\big(d\log(2n_{1:K}) + \log(1/\delta)\big)} \\ & + 2\min_h \left\{ e_T(h) + \sum_{i=1}^{K} \alpha_i e_i(h) \right\} + d_{\mathcal{H}\Delta\mathcal{H}}(D_{\boldsymbol{\alpha}}, D_T) \ . \end{aligned}$$

2.9 Open Questions

There are a variety of open technical problems related to this work. For example, Theorems 5 and 6 provide bounds on the error of a classifier found by minimizing a weighted empirical error term for a fixed weight vector $\boldsymbol{\alpha}$. However, we do not have a general efficient algorithm for finding the weight vector $\boldsymbol{\alpha}$ that minimizes the bound. It is also not known how these bounds will change if the VC dimension is replaced by a more sophisticated measure of complexity as in Theorem 3, though we believe it should be possible to derive analogous results.

More broadly, as mentioned above, the results described in this chapter can be viewed as a simple theoretical foundation for rudimentary collaborative filtering. While there have been some recent theoretical advancements in collaborative filtering (largely focused on techniques for low-rank matrix completion), there is still a wide gap between theory and practice. One of the hardest aspects of closing this gap is developing the right model for the problem; the simple models discussed in this chapter are not powerful enough to yield general practical algorithms, but the popular matrix-completion models tend to discard valuable feature information that may be extremely important when there is only a small amount of training data available. The problem of developing better models for collaborative filtering and some of the difficulties that arise are discussed in Chapter 6.

Chapter 3

Learning from Collective Behavior

Collective behavior in large populations has been a subject of enduring interest in sociology and economics, and has recently gained momentum in computer science. There is consequently now an impressive literature on mathematical models of collective behavior for phenomena as diverse as the diffusion of fads in social networks [58, 86, 132], voting behavior [58], housing choices and segregation [113], herding behaviors in financial markets [131], information propagation on blogs [96, 60], the transmission of infectious diseases or computer viruses [46, 18], the prevention of water contamination [95], the spread of new agricultural or medical practices [110, 36], Hollywood trends [43], and the contagion properties of obesity [33]. The growing popularity of the Internet has greatly increased the number of both controlled experiments [47, 49, 80, 111, 112, 83] and open-ended systems (such as Wikipedia and many other instances of "human peer-production") that permit the logging and analysis of detailed collective behavioral data. It is natural to ask if there are learning methods specifically tailored to such models and data.

In this chapter, we introduce a computational theory of learning from collective behavior, in which the goal is to accurately model and predict the future behavior of a large population after observing their interactions during a training phase of polynomial length. We define a formal model for efficient learning in such settings, and develop general theory for this model, including a polynomial-time reduction of learning from collective behavior to more traditional i.i.d. learning. We define specific classes of agent strategies, including "crowd affinity" and complementary "crowd aversion" classes, and provide provably efficient algorithms for learning from collective behavior for these classes.

This material is based on joint work with Michael Kearns [77].

3.1 Overview

For decades, collective behavior has been a subject of interest in sociology and economics. The specific mathematical models found throughout the collective behavior literature differ from one another in terms of technical details, but generally share the significant underlying assumption that each agent's current behavior is entirely or largely determined by the recent behavior of the other agents. Thus the collective behavior is a *social* phenomenon. That is, the population evolves over time according to its own internal dynamics. There is generally no exogenous "Nature" being reacted to or injecting shocks to the collective.

In this chapter, we provide a new computational theory of learning from collective behavior. We assume that each agent i in a population of size N acts according to a fixed but unknown strategy c_i drawn from a known class $\mathcal{C}$. A strategy probabilistically maps the current population state to the next state or action for that agent, and each agent's strategy may be different. As is common in much of the literature, there may also be a network structure governing the population interaction, in which case strategies may map the local neighborhood state to next actions.

Learning algorithms in our model are given training data of the population behavior, either as repeated finite-length trajectories from multiple initial states (an *episodic* model), or in a single unbroken trajectory from a fixed start state (a *no-reset* model). In either case, they must efficiently (polynomially) learn to accurately predict or simulate (properties of) the future behavior of the same population. Our framework may be viewed as a computational model for learning the dynamics of an unknown Markov process — more precisely, a dynamic Bayes net — in which our primary interest is in Markov processes inspired by simple models for social behavior. The relationship between our results and prior work on parameter learning in Bayesian networks is discussed in Section 3.2.2.

As a simple, concrete example of the kind of system we have in mind, consider a population in which each agent makes a series of choices from a fixed set over time (such as what restaurant to go to, or what political party to vote for). Like many previously studied models, we consider agents who have a desire to behave like the rest of the population (because they want to visit the popular

restaurants, or want to vote for "electable" candidates). On the other hand, each agent may also have different and unknown intrinsic preferences over the choices as well (based on cuisine and decor, or the actual policies of the candidates). We consider models in which each agent balances or integrates these two forces in deciding how to behave at each step [66]. Our main question is: Can a learning algorithm watching the collective behavior of such a population for a short period produce an accurate model of their future choices?

The assumptions of our model fit nicely with the literature cited above, much of which indeed proposes simple stochastic models for how individual agents react to the current population state. We emphasize from the outset the difference between our interests and those common in multiagent systems and learning in games. In those fields, it is often the case that the agents themselves are acting according to complex and fairly general learning algorithms (such as Q-learning [130], no-regret learning [54], or fictitious play [23]). In contrast, while the agent strategies we consider are certainly "adaptive" in a reactive sense, they are much simpler than general-purpose learning algorithms, and we are interested in learning algorithms that *model* the full collective behavior. Thus our interest is not in learning by the agents themselves, but at the higher level of an observer of the population.

The primary contributions described in this chapter are the introduction of a computational model for learning from collective behavior, the development of some general theory for this model, the definition of specific classes of agent strategies, including variants of the "crowd affinity" strategies sketched above, and complementary "crowd aversion" classes, and provably efficient algorithms for learning from collective behavior for these same classes.

Chapter Outline: In Section 3.2, we introduce our main model for learning from collective behavior and discuss two natural variants. Section 3.3 introduces and motivates a number of specific agent strategy classes that are broadly inspired by earlier sociological models and provides brief simulations of the collective behaviors they can generate. Section 3.4 provides a general reduction of learning from collective behavior to a generalized PAC-style model for learning from i.i.d. data, which is used subsequently in Section 3.5, where we give provably efficient algorithms for learning some of the strategy classes introduced in Section 3.3. Brief conclusions and topics for further research are given in Section 3.6.

3.2 The Model

In this section we describe a learning model in which the observed data is generated from observations of trajectories (defined shortly) of the collective behavior of N interacting agents. The key feature of the model is the fact that each agent's next state or action is always *determined by the recent actions of the other agents*, perhaps combined with some intrinsic "preferences" or behaviors of the particular agent. As we shall see, we can view our model as one for learning certain kinds of factored Markov processes that are inspired by models common in sociology and related fields.

Each agent may follow a different and possibly probabilistic strategy. We assume that the strategy followed by each agent is constrained to lie in a known (and possibly large) class, but is otherwise unknown. The learner's ultimate goal is not to discover each individual agent strategy per se, but rather to make accurate predictions of the *collective* behavior in novel situations.

3.2.1 Agent Strategies and Collective Trajectories

The main components of our framework are as follows:

- **State Space.** At each time step, each agent i is in some state s_i chosen from a known, finite set $\mathcal{S}$ of size K. We often think of K as being large, and thus want algorithms whose running time scales polynomially in K and other parameters. We view s_i as the *action* taken by agent i in response to the recent population behavior. The joint action vector $\mathbf{s} \in \mathcal{S}^N$ describes the current global state of the collective.

- **Initial State Distribution.** We assume that the initial population state $\mathbf{s}^0$ is drawn according to a fixed but unknown distribution P over $\mathcal{S}^N$. During training, the learner is able to see trajectories of the collective behavior in which the initial state is drawn from P, and as in many standard learning models, must generalize with respect to this same distribution.

- **Agent Strategy Class.** We assume that each agent's strategy is drawn from a known class $\mathcal{C}$ of (typically probabilistic) mappings from the recent collective behavior into the agent's next state or action in $\mathcal{S}$. We mainly consider the case in which $c_i \in \mathcal{C}$ probabilistically maps the current global state $\mathbf{s}$ into agent i's next state. However, much of the theory we develop applies equally well to more complex strategies that might incorporate a longer history of the collective behavior on the current trajectory.

Given these components, we can now define what is meant by a *collective trajectory*.

Definition 3 *Let $\mathbf{c} \in \mathcal{C}^N$ be the vector of strategies for the N agents, P be the initial state distribution, and $T \geq 1$ be an integer. A T-***trajectory of $\mathbf{c}$ with respect to** P *is a random variable $\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle$ in which the initial state $\mathbf{s}^0 \in \mathcal{S}^N$ is drawn according to P, and for each $t \in \{1, \cdots, T\}$, the component s_i^t of the joint state $\mathbf{s}^t$ is obtained by applying the strategy c_i to $\mathbf{s}^{t-1}$.*

Thus, a collective trajectory in our model is simply a Markovian sequence of states that *factors* according to the N agent strategies — that is, a dynamic Bayes net [57]. Our interest is in cases in which this Markov process is generated by particular models of social behavior.

3.2.2 The Learning Model

We now formally define the learning model we study. In our model, learning algorithms are given access to an oracle $\mathcal{O}_{\mathrm{EXP}}(\mathbf{c}, P, T)$ that returns a T-trajectory $\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle$ of $\mathbf{c}$ with respect to P. This is thus an *episodic* or *reset* model, in which the learner has the luxury of repeatedly observing the population behavior from random initial conditions. It is most applicable in (partially) controlled, experimental settings [47, 49, 80, 111, 112, 83] where such "population resets" can be implemented or imposed. In Section 3.2.3 below we define a perhaps more broadly applicable variant of the model in which resets are not available; the algorithms we provide can be adapted for this model as well (see Appendix A3.8).

The goal of the learner is to find a *generative model* that can efficiently produce trajectories from a distribution that is arbitrarily close to that generated by the true population. Thus, let $\hat{M}(\mathbf{s}^0, T)$ be a (randomized) model output by a learning algorithm that takes as input a start state $\mathbf{s}^0$ and time horizon T, and outputs a random T-trajectory, and let $Q_{\hat{M}}$ denote the distribution over trajectories generated by $\hat{M}$ when the start state is distributed according to P. Similarly, let $Q_{\mathbf{c}}$ denote the distribution over trajectories generated by $\mathcal{O}_{\mathrm{EXP}}(\mathbf{c}, P, T)$. Then the goal of the learning algorithm is to find a model $\hat{M}$ making the $\mathcal{L}_1$ distance $\varepsilon(Q_{\hat{M}}, Q_{\mathbf{c}})$ between $Q_{\hat{M}}$ and $Q_{\mathbf{c}}$ small, where

$$\varepsilon(Q_{\hat{M}}, Q_{\mathbf{c}}) \equiv \sum_{\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle} \left| Q_{\hat{M}}(\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle) - Q_{\mathbf{c}}(\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle) \right| .$$

Note that we have defined the output of the learning algorithm to be a "black box" that simply produces trajectories from initial states. Of course, it would be natural to expect that this black box operates by having good approximations to every agent strategy in $\mathbf{c}$, and using collective simulations of these to produce trajectories, but we choose to define the output $\hat{M}$ in a more general way since there may be other approaches. Second, we note that our learning criteria is both strong (see below for a discussion of weaker alternatives) and useful, in the sense that if $\varepsilon(Q_{\hat{M}}, Q_{\mathbf{c}})$ is smaller than ϵ, then we can sample $\hat{M}$ to obtain $O(\epsilon)$-good approximations to the expectation of any (bounded) *function* of trajectories. Thus, for instance, we can use $\hat{M}$ to answer questions like "What is the expected number of agents playing the plurality action after T steps?" or "What is the probability the entire population is playing the same action after T steps?"

Our algorithmic results consider cases in which the agent strategies may themselves already be rather rich, in which case the learning algorithm should be permitted resources commensurate with this complexity. For example, the crowd affinity models have a number of parameters that scales with the number of actions K. More generally, we use $dim(\mathcal{C})$ to denote the complexity or dimension of $\mathcal{C}$; in all of our imagined applications $dim(\cdot)$ is either the VC dimension for deterministic classes, or one of its generalizations to probabilistic classes (such as pseudo-dimension [65], fat-shattering dimension [74], combinatorial dimension [65], etc.).

We are now ready to define our learning model.

Definition 4 *Let $\mathcal{C}$ be an agent strategy class over actions $\mathcal{S}$. We say that $\mathcal{C}$ is* **polynomially learnable from collective behavior** *if there exists an algorithm A such that for any population size $N \geq 1$, any $\mathbf{c} \in \mathcal{C}^N$, any time horizon T, any distribution P over $\mathcal{S}^N$, and any $\epsilon > 0$ and $\delta > 0$, given access to the oracle $\mathcal{O}_{\mathrm{EXP}}(\mathbf{c}, P, T)$, algorithm A runs in time polynomial in N, T, $dim(\mathcal{C})$, $1/\epsilon$, and $1/\delta$, and outputs a polynomial-time model $\hat{M}$ such that with probability at least $1 - \delta$, $\varepsilon(Q_{\hat{M}}, Q_{\mathbf{c}}) \leq \epsilon$.*

Note that if we cared only about sample complexity and not efficiency, we could apply Dasgupta's results on parameter learning in Bayesian networks [42] to show that any class $\mathcal{C}$ with finite pseudo-dimension $dim(\mathcal{C})$ is learnable from collective behavior from a number of samples polynomial in N, T, $dim(\mathcal{C})$, $1/\epsilon$, and $1/\delta$. However, Dasgupta's analysis involves directly bounding the

pseudo-dimension of the class of Bayesian networks and therefore does not provide any algorithmic insight. More recently, Abbeel et al. [1] showed that efficient parameter learning in Bayesian networks is possible when nodes have bounded in-degree. Since the maximum in-degree in our setting is N, their techniques would lead to an exponential dependence on the number of agents in both the run time and sample complexity.

On the other hand, the guarantees of both Dasgupta and Abbeel et al. are stronger than ours in the sense that they require low KL divergence between the true joint distribution and the joint distribution induced by the model while we ask for only low $\mathcal{L}_1$ distance. For Abbeel et al., this results in an inverse dependence in the sample complexity on the smallest conditional probability in the network, which we avoid. Dasgupta gets around this problem by explicitly restricting the space of models to those in which all probabilities are bounded away from zero.

We now discuss two reasonable variations on the model we have presented.

3.2.3 A No-Reset Variant

The model above assumes that learning algorithms are given access to repeated, independent trajectories via the oracle $\mathcal{O}_{\mathrm{EXP}}$, which is analogous to the *episodic* setting of reinforcement learning. As in that field, we may also wish to consider an alternative "no-reset" model in which the learner has access only to a *single*, unbroken trajectory of states generated by the Markov process. To do so we must formulate an alternative notion of generalization, since on the one hand, the (distribution of the) initial state may quickly become irrelevant as the collective behavior evolves, but on the other, the state space is exponentially large and thus it is unrealistic to expect to model the dynamics from an *arbitrary* state in polynomial time.

One natural formulation allows the learner to observe any polynomially long prefix of a trajectory of states for training, and then to announce its readiness for the test phase. If $\mathbf{s}$ is the final state of the training prefix, we can simply ask that the learner output a model $\hat{M}$ that generates accurate T-step trajectories *forward* from the current state $\mathbf{s}$. In other words, $\hat{M}$ should generate trajectories from a distribution close to the distribution over T-step trajectories that would be generated if each agent continued choosing actions according to his strategy. The length of the training prefix is allowed to be polynomial in T and the other parameters.

While aspects of the general theory described below are particular to our main (episodic)

model, we note here that the algorithms we give for specific classes can in fact be adapted to work in the no-reset model as well. Such extensions are discussed in Appendix A3.8.

3.2.4 Weaker Criteria for Learnability

We have chosen to formulate learnability in our model using a rather strong success criterion — namely, the ability to (approximately) simulate the full dynamics of the unknown Markov process induced by the population strategy $\mathbf{c}$. In order to meet this strong criterion, we have also allowed the learner access to a rather strong oracle, which returns all *intermediate* states of sampled trajectories.

There may be natural scenarios, however, in which we are interested only in specific *fixed* properties of collective behavior, and thus a weaker data source may suffice. For instance, suppose we have a fixed, real-valued *outcome function* $F(\mathbf{s}^T)$ of final states (for instance, the fraction of agents playing the plurality action at time T), with our goal being to simply learn a function G that maps initial states $\mathbf{s}^0$ and a time horizon T to real values, and approximately minimizes

$$\mathrm{E}_{\mathbf{s}^0 \sim P}\left[\left|G(\mathbf{s}^0, T) - \mathrm{E}_{\mathbf{s}^T}\left[F(\mathbf{s}^T)\right]\right|\right]$$

where $\mathbf{s}^T$ is a random variable that is the final state of a T-trajectory of $\mathbf{c}$ from the initial state $\mathbf{s}^0$. Clearly in such a model, while it certainly would suffice, there may be no need to directly learn a full dynamical model. It may be feasible to satisfy this criterion without even observing intermediate states, but only seeing initial state and final outcome pairs $\langle \mathbf{s}^0, F(\mathbf{s}^T)\rangle$, closer to a traditional regression problem.

It is not difficult to define simple agent strategy classes for which learning from only $\langle \mathbf{s}^0, F(\mathbf{s}^T)\rangle$ pairs is provably intractable, yet efficient learning is possible in our model. This idea is formalized in Theorem 7 below. The idea behind the proof is that the population forms a rather powerful computational device mapping initial states to final states. In particular, it can be thought of as a circuit of depth T with "gates" chosen from $\mathcal{C}$, with the only real constraint being that each layer of the circuit is an identical sequence of N gates which are applied to the outputs of the previous layer. Intuitively, if only initial states and final outcomes are provided to the learner, learning should be as difficult as a corresponding PAC-style problem. On the other hand, by observing intermediate state vectors we can build arbitrarily accurate models for each agent, which

in turn allows us to accurately simulate the full dynamical model. A sketch of the full proof is in Appendix A3.1.

Theorem 7 *Let $\mathcal{C}$ be the class of 2-input* AND *and* OR *gates, and one-input* NOT *gates. Then $\mathcal{C}$ is polynomially learnable from collective behavior, but there exists a binary outcome function F such that learning an accurate mapping from start states $\mathbf{s}^0$ to outcomes $F(\mathbf{s}^T)$ without observing intermediate state data is intractable.*

Conversely, it is also not difficult to concoct cases in which learning the full dynamics in our sense is intractable, but we can learn to approximate a specific outcome function from only $\langle \mathbf{s}^0, F(\mathbf{s}^T) \rangle$ pairs. Intuitively, if each agent strategy is very complex but the outcome function applied to final states is sufficiently simple (e.g., constant), we cannot but do not need to model the full dynamics in order to learn to approximate the outcome.

We note that there is an analogy here to the distinction between *direct* and *indirect* approaches to reinforcement learning [75]. In the former, one learns a policy that is specific to a fixed reward function without learning a model of next-state dynamics; in the latter, at possibly greater cost, one learns an accurate dynamical model, which can in turn be used to compute good policies for any reward function. For the remainder of this chapter, we focus on the model as we formalized it in Definition 4, and leave for future work the investigation of such alternatives.

3.3 Social Strategy Classes

Before providing our general theory, including the reduction from collective learning to i.i.d. learning, we first illustrate and motivate the definitions so far with some concrete examples of social strategy classes, some of which we analyze in detail in Section 3.5.

3.3.1 Crowd Affinity: Mixture Strategies

The first class of agent strategies we discuss are meant to model settings in which each individual wishes to balance their intrinsic personal preferences with a desire to "follow the crowd." We broadly refer to strategies of this type as *crowd affinity* strategies (in contrast to the *crowd aversion* strategies discussed shortly), and examine a couple of natural variants.

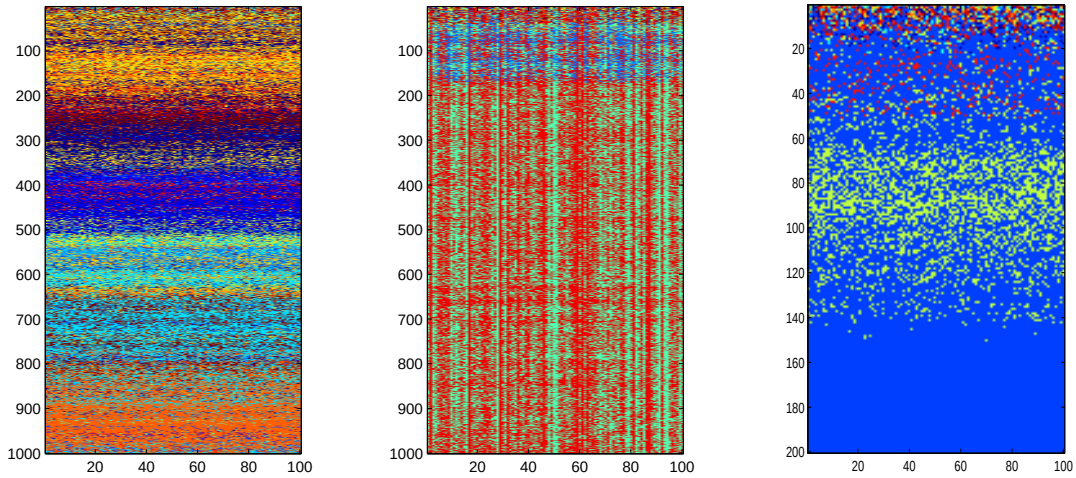
As a motivating example, imagine that there are K restaurants, and each week, every member of a population chooses one of the restaurants in which to dine. On the one hand, each agent has personal preferences over the restaurants based on the cuisine, service, ambiance, and so on. On the other, each agent has some desire to go to the currently "hot" restaurants — that is, where many or most other agents have been recently. To model this setting, let $\mathcal{S}$ be the set of K restaurants, and suppose $\mathbf{s} \in \mathcal{S}^N$ is the population state vector indicating where each agent dined last week. We can summarize the population behavior by the vector or distribution $\mathbf{f} \in [0,1]^K$, where f_a is the fraction of agents dining in restaurant a in $\mathbf{s}$. Similarly, we might represent the personal preferences of a specific agent by another distribution $\mathbf{w} \in [0,1]^K$ in which w_a represents the probability this agent would attend restaurant a in the absence of any information about what the population is doing. One natural way for the agent to balance their preferences with the population behavior would be to choose a restaurant according to the mixture distribution $(1-\alpha)\mathbf{f} + \alpha\mathbf{w}$ for some agent-dependent mixture coefficient α. Such models have been studied in the sociology literature [66] in the context of belief formation.

We are interested in collective systems in which every agent i has some unknown preferences $\mathbf{w}_i$ and mixture coefficient α_i, and in each week t chooses its next restaurant according to $(1-\alpha_i)\mathbf{f}^t + \alpha_i\mathbf{w}_i$, which thus probabilistically yields the next population distribution $\mathbf{f}^{t+1}$. How do such systems behave? And how can we learn to model their macroscopic properties from only observed behavior, especially when the number of choices K is large?

An illustration of the rich collective behavior that can already be generated from such simple strategies is shown in Figure 3.1(a). Here we show a single but typical 1000-step simulation of collective behavior under this model, in which $N = 100$ and each agent's individual preference vector $\mathbf{w}$ puts all of its weight on just one of 10 possible actions (represented as different shades of gray, or colors in the full-color version); this action was selected independently at random for each agent. All agents have an α value of just 0.01, and thus are selecting from the population distribution 99% of the time. Each row shows the population state at a given step, with time increasing down the horizontal axis of the image. The initial state was chosen uniformly at random.

It is interesting to note the dramatic difference between $\alpha = 0$ (in which rapid convergence to a common color is certain) and this small value for α; despite the fact that almost all agents play the population distribution at every step, revolving horizontal waves of near-consensus to different

Figure 3.1 Sample simulations of a) the crowd affinity mixture model, b) the crowd affinity multiplicative model, and c) the agent affinity model. See the text for details.



choices are present, with no final convergence in sight. The slight "personalization" of population-only behavior is enough to dramatically change the collective behavior. Broadly speaking, it is such properties we would like a learning algorithm to model and predict from sufficient observations.

3.3.2 Crowd Affinity: Multiplicative Strategies

One possible objection to the crowd affinity mixture strategies described above is that each agent can be viewed as *randomly* choosing whether to *entirely* follow the population distribution (with probability $1 - \alpha$) or to *entirely* follow their personal preferences (with probability α) at each time step. A more realistic model might have each agent truly *combine* the population behavior with their preferences at every step.

Consider, for instance, how an American citizen might alter their anticipated presidential voting decision over time in response to recent primary or polling news. If their first choice of candidate — say, an Independent or Libertarian candidate — appears over time to be "unelectable" in the general election due to their inability to sway large numbers of Democratic and Republican voters, a natural and typical response is for the citizen to shift their intended vote to whichever of the front-runners they most prefer or least dislike. In other words, the low popularity of their first choice causes that choice to be dampened or eradicated; unlike the mixture model above, where weight α

is always given to personal preferences, here there may remain *no* weight on this candidate.

One natural way of defining a general such class of strategies is as follows. As above, let $\mathbf{f} \in [0,1]^K$, where f_a is the fraction of agents dining in restaurant a in the current state $\mathbf{s}$. Similar to the mixture strategies above, let $\mathbf{w}_i \in [0,1]^K$ be a vector of *weights* representing the intrinsic preferences of agent i over actions. Then define the probability that agent i plays action a to be $f_a \cdot w_{i,a}/Z(\mathbf{f}, \mathbf{w}_i)$, where the normalizing factor is $Z(\mathbf{f}, \mathbf{w}_i) = \sum_{b \in \mathcal{S}} f_b \cdot w_{i,b}$. Thus, in such *multiplicative* crowd affinity models, the probability the agent takes an action is always proportional to the product of their preference for it and its current popularity.

Despite their similar motivation, the mixture and multiplicative crowd affinity strategies can lead to dramatically different collective behavior. Perhaps the most obvious difference is that in the mixture case, if agent i has a strong preference for action a there is *always* some minimum probability ($\alpha_i w_{i,a}$) they take this action, whereas in the multiplicative case even a strong preference can be eradicated from expression by small or zero values for the popularity f_a.

In Figure 3.1(b), we again show a single but typical 1000-step, $N = 100$ simulation for the multiplicative model in which agent's individual preference distributions $\mathbf{w}$ are chosen to be random normalized vectors over 10 actions. The dynamics are now quite different than for the additive crowd affinity model. In particular, now there is never near-consensus but a gradual dwindling of the shades or colors represented in the population — from the initial full diversity down to 3 colors remaining at approximately $t = 100$, until by $t = 200$ there is a stand-off in the population between red and light green. Unlike the additive models, colors die out in the population permanently. There is also clear vertical structure corresponding to strong conditional preferences of the agents once the stand-off emerges.

3.3.3 Crowd Aversion and Other Variants

It is easy to transform the mixture or multiplicative crowd affinity strategies into *crowd aversion* strategies — that is, in which agents wish to balance or combine their personal preferences with a desire to act *differently* than the population at large. This can be accomplished in a variety of simple ways. For instance, if $\mathbf{f}$ is the current distributions over actions in the population, we can simply define a kind of "inverse" to the distribution by letting $g_a = (1 - f_a)/(K - 1)$, where $K - 1 = \sum_{b \in \mathcal{S}}(1 - f_b)$ is the normalizing factor, and applying the strategies above to $\mathbf{g}$ rather

than $\mathbf{f}$. Now each agent exhibits a tendency to "avoid the crowd", moderated as before by their own preferences.

Of course, there is no reason to assume that the entire population is crowd-seeking, or crowd-avoiding; more generally we would allow there to be both types of individuals present. Furthermore, we might entertain other transforms of the population distribution than just g_a above. For instance, we might wish to still consider crowd affinity, but to first "sharpen" the distribution by replacing each f_a with f_a^2 and normalizing, then applying the models discussed above to the resulting vector. This has the effect of magnifying the attraction to the most popular actions. In general our algorithmic results are robust to a wide range of such variations.

3.3.4 Agent Affinity and Aversion Strategies

In the two versions of crowd affinity strategies discussed above, an agent has personal preferences over actions, and also reacts to the current population behavior, but only in an aggregate fashion. An alternative class of strategies that we call *agent affinity* strategies instead allows agents to prefer to agree (or disagree) in their choice with specific other agents.

For a fixed agent, such a strategy can be modeled by a weight vector $\mathbf{w} \in [0,1]^N$, with one weight for each *agent* in the population rather than each action. We define the probability that this agent takes action a if the current global state is $\mathbf{s} \in \mathcal{S}^N$ to be proportional to $\sum_{i:s_i=a} w_i$. In this class of strategies, the strength of the agent's desire to take the same action as agent i is determined by how large the weight w_i is. The overall behavior of this agent is then probabilistically determined by summing over all agents in the fashion above.

In Figure 3.1(c), we show a single but typical simulation, again with $N = 100$ but now with a much shorter time horizon of 200 steps and a much larger set of 100 actions. All agents have random distributions as their preferences over other agents; this model is similar to traditional diffusion dynamics in a dense, random (weighted) network, and quickly converges to global consensus.

We leave the analysis of this strategy class to future work, but remark that in the simple case in which $K = 2$, learning this class is closely related to the problem of learning perceptrons under certain noise models in which the intensity of the noise increases with proximity to the separator [35, 24] and seems at least as difficult.

3.3.5 Incorporating Network Structure

Many of the social models inspiring this work involve a network structure that dictates or restricts the interactions between agents [86]. It is natural to ask if the strategy classes discussed here can be extended to the scenario in which each agent is influenced only by his neighbors in a given network. Indeed, it is straightforward to extend each of the strategy classes introduced in this section to a network setting. For example, to adapt the crowd affinity and aversion strategy classes, it suffices to redefine f_a for each agent i to be the fraction of agents in the local neighborhood of agent i choosing action a. To adapt the agent affinity and aversion classes, it is necessary only to require that $w_j = 0$ for every agent j outside the local neighborhood of agent i. By making these simple modifications, the learning algorithms discussed in Section 3.5 can immediately be applied to settings in which a network structure is given.

3.4 A Reduction to I.I.D. Learning

Since algorithms in our framework are attempting to learn to model the dynamics of a factored Markov process in which each component is known to lie in the class $\mathcal{C}$, it is natural to investigate the relationship between learning just a single strategy in $\mathcal{C}$ and the entire Markovian dynamics. One main concern might be effects of dynamic instability — that is, that even small errors in models for each of the N components could be amplified exponentially in the overall population model. In this section we show that this can be avoided. More precisely, we prove that if the component errors are all small compared to $1/(NT)^2$, the population model also has small error. Thus fast rates of learning for individual components are polynomially preserved in the resulting population model.

To show this, we give a reduction showing that if a class $\mathcal{C}$ of (possibly probabilistic) strategies is polynomially learnable (in a sense that we describe shortly) from i.i.d. data, then $\mathcal{C}$ is also polynomially learnable from collective behavior. The key step in the reduction is the introduction of the experimental distribution, defined below. Intuitively, the experimental distribution is meant to capture the distribution over states that are encountered in the collective setting over repeated trials. Polynomial i.i.d. learning on this distribution leads to polynomial learning from the collective.

3.4.1 A Reduction for Deterministic Strategies

In order to illustrate some of the key ideas we use in the more general reduction, we begin by examining the simple case in which the number of actions $K = 2$ and and each strategy $c \in \mathcal{C}$ is deterministic. We show that if $\mathcal{C}$ is polynomially learnable in the (distribution-free) PAC model, then $\mathcal{C}$ is polynomially learnable from collective behavior.

In order to exploit the fact that $\mathcal{C}$ is PAC learnable, it is first necessary to define a single distribution over states on which we would like to learn.

Definition 5 *For any initial state distribution P, strategy vector $\mathbf{c}$, and sequence length T, the* **experimental distribution** *$D_{P,\mathbf{c},T}$ is the distribution over state vectors $\mathbf{s}$ obtained by querying $\mathcal{O}_{\mathrm{EXP}}(\mathbf{c}, P, T)$ to obtain $\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle$, choosing t uniformly at random from $\{0, \cdots, T-1\}$, and setting $\mathbf{s} = \mathbf{s}^t$.*

We denote this distribution simply as D when P, $\mathbf{c}$, and T are clear from context. Given access to the oracle $\mathcal{O}_{\mathrm{EXP}}$, we can sample pairs $\langle \mathbf{s}, c_i(\mathbf{s}) \rangle$ where $\mathbf{s}$ is distributed according to D using the following procedure:

1. Query $\mathcal{O}_{\mathrm{EXP}}(\mathbf{c}, P, T)$ to obtain $\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle$.
2. Choose $t \in \{0, \cdots, T-1\}$ uniformly at random.
3. Return $\left\langle \mathbf{s}^t, s_i^{t+1} \right\rangle$.

If $\mathcal{C}$ is polynomially learnable in the PAC model, then by definition, with access to the oracle $\mathcal{O}_{\mathrm{EXP}}$, for any $\delta, \epsilon > 0$, it is possible to learn a model $\hat{c}_i$ such that with probability $1 - (\delta/N)$,

$$\Pr_{\mathbf{s} \sim D} \left[\hat{c}_i(\mathbf{s}) \neq c_i(\mathbf{s}) \right] \leq \frac{\epsilon}{NT}$$

in time polynomial in N, T, $1/\epsilon$, $1/\delta$, and the VC dimension of $\mathcal{C}$ using the sampling procedure above; the dependence on N and T come from the fact that we are requesting a confidence of $1 - (\delta/N)$ and an accuracy of $\epsilon/(TN)$. We can learn a set of such strategies $\hat{c}_i$ for all agents i at the cost of an additional factor of N.

Consider a new sequence $\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle$ returned by the oracle $\mathcal{O}_{\mathrm{EXP}}$. By the union bound, with probability $1 - \delta$, the probability that there exists any agent i and any $t \in \{0, \cdots, T-1\}$, such

that $\hat{c}_i(\mathbf{s}^t) \neq c_i(\mathbf{s}^t)$ is less than ϵ. If this is not the case (i.e., if $\hat{c}_i(\mathbf{s}^t) = c_i(\mathbf{s}^t)$ for all i and t) then the same sequence of states would have been reached if we had instead started at state $\mathbf{s}^0$ and generated each additional state $\mathbf{s}^t$ by letting $s_i^t = c_i(\mathbf{s}^{t-1})$. This implies that with probability $1-\delta$, $\varepsilon(Q_{\hat{M}}, Q_{\mathbf{c}}) \leq \epsilon$, and $\mathcal{C}$ is polynomially learnable from collective behavior.

3.4.2 A General Reduction

Multiple analogs of the definition of learnability in the PAC model have been proposed for distribution learning settings. The probabilistic concept model [74] presents a definition for learning conditional distributions over binary outcomes, while later work [78] proposes a definition for learning unconditional distributions over larger outcome spaces. We combine the two into a single PAC-style model for learning conditional distributions over large outcome spaces from i.i.d. data as follows.

Definition 6 *Let $\mathcal{C}$ be a class of probabilistic mappings from an input $\mathbf{x} \in \mathcal{X}$ to an output $y \in \mathcal{Y}$ where $\mathcal{Y}$ is a finite set. We say that $\mathcal{C}$ is* **polynomially learnable** *if there exists an algorithm A such that for any $c \in \mathcal{C}$ and any distribution D over $\mathcal{X}$, if A is given access to an oracle producing pairs $\langle \mathbf{x}, c(\mathbf{x}) \rangle$ with x distributed according to D, then for any $\epsilon, \delta > 0$, algorithm A runs in time polynomial in $1/\epsilon$, $1/\delta$, and $dim(\mathcal{C})$ and outputs a function $\hat{c}$ such that with probability $1-\delta$,*

$$\mathrm{E}_{\mathbf{x}\sim D}\left[\sum_{y\in\mathcal{Y}} \left|\Pr\left[c(\mathbf{x}) = y\right] - \Pr\left[\hat{c}(\mathbf{x}) = y\right]\right|\right] \leq \epsilon\ .$$

We could have chosen instead to require that the expected KL divergence between c and $\hat{c}$ be bounded. Using Jensen's inequality and Lemma 12.6.1 of Cover and Thomas [38], it is simple to show that if the expected KL divergence between two distributions is bounded by ϵ, then the expected $\mathcal{L}_1$ distance is bounded by $\sqrt{2\ln(2)\epsilon}$. Thus any class that is polynomially learnable under this alternate definition is also polynomially learnable under ours.

Theorem 8 *For any class $\mathcal{C}$, if $\mathcal{C}$ is polynomially learnable according to Definition 6, then $\mathcal{C}$ is polynomially learnable from collective behavior.*

Proof: This proof is very similar in spirit to the proof of the reduction for the deterministic case. However, several tricks are needed to deal with the fact that trajectories are now random variables,

even given a fixed start state. In particular, it is no longer the case that we can argue that starting at a given start state and executing a set of strategies that are "close to" the true strategy vector usually yields *the same* full trajectory we would have obtained by executing the true strategies of each agent. Instead, due to the inherent randomness in the strategies, we must argue that the *distribution* over trajectories is similar when the estimated strategies are sufficiently close to the true strategies.

To make this argument, we begin by introducing the idea of sampling from a distribution P_1 using a "filtered" version of a second distribution P_2 as follows. First, draw an outcome $\omega \in \Omega$ according to P_2. If $P_1(\omega) \geq P_2(\omega)$, output ω. Otherwise, output ω with probability $P_1(\omega)/P_2(\omega)$, and with probability $1 - P_1(\omega)/P_2(\omega)$, output an alternate action drawn according to a third distribution P_3, where

$$P_3(\omega) = \frac{P_1(\omega) - P_2(\omega)}{\sum_{\omega' : P_2(\omega') < P_1(\omega')} P_1(\omega') - P_2(\omega')}$$

if $P_1(\omega) > P_2(\omega)$, and $P_3(\omega) = 0$ otherwise.

It is easy to verify that the output of this filtering algorithm is indeed distributed according to P_1. Additionally, notice that the probability that the output is "filtered" is

$$\sum_{\omega : P_2(\omega) > P_1(\omega)} P_2(\omega) \left(1 - \frac{P_1(\omega)}{P_2(\omega)}\right) = \frac{1}{2} ||P_2 - P_1||_1 \,. \tag{3.1}$$

As in the deterministic case, we make use of the experimental distribution D as defined in Definition 5. If $\mathcal{C}$ is polynomially learnable as in Definition 6, then with access to the oracle $\mathcal{O}_{\mathrm{EXP}}$, for any $\delta, \epsilon > 0$, it is possible to learn a model $\hat{c}_i$ such that with probability $1 - (\delta/N)$,

$$\mathrm{E}_{\mathbf{s} \sim D} \left[\sum_{s \in \mathcal{S}} |\Pr\left[c_i(\mathbf{s}) = s\right] - \Pr\left[\hat{c}_i(\mathbf{s}) = s\right]| \right] \leq \left(\frac{\epsilon}{NT}\right)^2 \tag{3.2}$$

in time polynomial in N, T, $1/\epsilon$, $1/\delta$, and $dim(\mathcal{C})$ using the three-step sampling procedure described in the deterministic case; as before, the dependence on N and T stem from the fact that we are requesting a confidence of $1 - (\delta/N)$ and an accuracy that is polynomial in both N and T. It is possible learn a set of such strategies $\hat{c}_i$ for all agents i at the cost of an additional factor of N.

If Equation 3.2 is satisfied for agent i, then for any $\tau \geq 1$, the probability of drawing a state $\mathbf{s}$ from D such that

$$\sum_{s \in \mathcal{S}} |\Pr[c_i(\mathbf{s}) = s] - \Pr[\hat{c}_i(\mathbf{s}) = s]| \geq \tau \left(\frac{\epsilon}{NT}\right)^2 \tag{3.3}$$

is no more than $1/\tau$.

Consider a new sequence $\langle \mathbf{s}^0, \cdots, \mathbf{s}^T \rangle$ returned by the oracle $\mathcal{O}_{\text{EXP}}$. For each $\mathbf{s}^t$, consider the action s_i^{t+1} chosen by agent i. This action was chosen according to the distribution c_i. Suppose instead we would like to choose this action according to the distribution $\hat{c}_i$ using a filtered version of c_i as described above. By Equation 3.1, the probability that the action choice of c_i is "filtered" (and thus not equal to s_i^{t+1}) is half the $\mathcal{L}_1$ distance between $c_i(\mathbf{s}^t)$ and $\hat{c}_i(\mathbf{s}^t)$. From Equation 3.3, we know that for any $\tau \geq 1$, with probability at least $1 - 1/\tau$, this probability is less than $\tau(\epsilon/(NT))^2$, so the probability of the new action being different from s_i^{t+1} is less than $\tau(\epsilon/(NT))^2 + 1/\tau$. This is minimized when $\tau = 2NT/\epsilon$, giving us a bound of $\epsilon/(NT)$.

By the union bound, with probability $1 - \delta$, the probability that there exists any agent i and any $t \in \{1, \cdots, T\}$, such that s_i^{t+1} is not equal to the action we get by sampling $\hat{c}_i(\mathbf{s}^t)$ using the filtered version of c_i must then be less than ϵ. As in the deterministic version, if this is *not* the case, then the same sequence of states would have been reached if we had instead started at state $\mathbf{s}^0$ and generated each additional state $\mathbf{s}^t$ by letting $s_i^t = \hat{c}_i(\mathbf{s}^{t-1})$ filtered using c_i. This implies that with probability $1 - \delta$, $\varepsilon(Q_{\hat{M}}, Q_{\mathbf{c}}) \leq \epsilon$, and $\mathcal{C}$ is polynomially learnable from collective behavior. ■

3.5 Learning Social Strategy Classes

We now turn our attention to efficient algorithms for learning some of the specific social strategy classes introduced in Section 3.3. We focus on the two crowd affinity model classes. Recall that these classes are designed to model the scenario in which each agent has an intrinsic set of preferences over actions, but simultaneously would prefer to choose the same actions chosen by other agents. Similar techniques can be applied to learn the crowd aversion strategies.

Formally, let $\mathbf{f}$ be a vector representing the distribution over current states of the agents; if $\mathbf{s}$ is the current state, then for each action a, $f_a = |\{i : s_i = a\}|/N$ is the fraction of the population currently choosing action a. (Alternately, if there is a network structure governing interaction among agents, f_a can be defined as the fraction of nodes in an agent's local neighborhood choosing

action a.) We denote by D^f the distribution over vectors $\mathbf{f}$ induced by the experimental distribution D over state vectors $\mathbf{s}$. In other words, the probability of a vector $\mathbf{f}$ under D^f is the sum over all state vectors $\mathbf{s}$ mapping to $\mathbf{f}$ of the probability of $\mathbf{s}$ under D.

We focus on the problem of learning the parameters of the strategy of a single agent i in each of the models. We assume that we are presented with a set of samples $\mathcal{M}$, where each instance $\mathcal{I}_m \in \mathcal{M}$ consists of a pair $\langle \mathbf{f}_m, a_m \rangle$. Here $\mathbf{f}_m$ is the distribution over states of the agents and a_m is the next action chosen by agent i. We assume that the state distributions $\mathbf{f}_m$ of these samples are distributed according to D^f. Given access to the oracle $\mathcal{O}_{\text{EXP}}$, such samples could be collected, for example, using a three-step procedure like the one in Section 3.4.1. We show that each class is polynomially learnable with respect to the distribution D^f induced by *any* distribution D over states, and so by Theorem 8, also polynomially learnable from collective behavior.

While it may seem wasteful to gather only one data instance for each agent i from each T-trajectory, we remark that only small, isolated pieces of the analysis presented in this section rely on the assumption that the state distributions of the samples are distributed according to D^f. In practice, the entire trajectories could be used for learning with no impact on the structure of the algorithms. Additionally, while the analysis here is geared towards learning under the experimental distribution, the algorithms we present can be applied without modification in the no-reset variant of the model; see Appendix A3.8.

3.5.1 Learning Crowd Affinity Mixture Models

In Section 3.3.1, we introduced the class of crowd affinity mixture model strategies. Such strategies are parameterized by a (normalized) weight vector $\mathbf{w}$ and parameter $\alpha \in [0, 1]$. The probability that agent i chooses action a given that the current state distribution is $\mathbf{f}$ is then $\alpha f_a + (1-\alpha)w_a$. In this section, we show that this class of strategies is polynomially learnable from collective behavior and sketch an algorithm for learning estimates of the parameters α and $\mathbf{w}$.

Let $\mathrm{I}(x)$ be the indicator function that is 1 if x is true and 0 otherwise. From the definition of the model it is easy to see that for any m such that $\mathcal{I}_m \in \mathcal{M}$, for any action $a \in \mathcal{S}$, $\mathrm{E}\left[\mathrm{I}(a_m = a)\right] = \alpha f_a + (1-\alpha)w_a$, where the expectation is over the randomness in the agent's strategy. By linearity

of expectation,

$$\mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}}\mathrm{I}(a_m=a)\right]=\alpha\sum_{m:\mathcal{I}_m\in\mathcal{M}}f_{m,a}+(1-\alpha)w_a|\mathcal{M}|\;. \tag{3.4}$$

Standard results from uniform convergence theory say that we can approximate the left-hand side of this equation arbitrarily well given a sufficiently large data set $\mathcal{M}$. Replacing the expectation with this approximation in Equation 3.4 yields a single equation with two unknown variables, α and w_a. To solve for these variables, we must construct a *pair* of equations with two unknown variables. We do so by splitting the data into instances where $f_{m,a}$ is "high" and instances where it is "low."

Specifically, let $M = |\mathcal{M}|$. For convenience of notation, assume without loss of generality that M is even; if M is odd, simply discard an instance at random. Define $\mathcal{M}_a^{low}$ to be the set containing the $M/2$ instances in $\mathcal{M}$ with the lowest values of $f_{m,a}$. Similarly, define $\mathcal{M}_a^{high}$ to be the set containing the $M/2$ instances with the highest values of $f_{m,a}$. Replacing $\mathcal{M}$ with $\mathcal{M}_a^{low}$ and $\mathcal{M}_a^{high}$ respectively in Equation 3.4 gives us two linear equations with two unknowns. As long as these two equations are linearly independent, we can solve the system of equations for α, giving us

$$\alpha=\frac{\mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{high}}\mathrm{I}(a_m=a)-\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{low}}\mathrm{I}(a_m=a)\right]}{\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{high}}f_{m,a}-\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{low}}f_{m,a}}.$$

We can approximate α from data in the natural way, using

$$\hat{\alpha}=\frac{\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{high}}\mathrm{I}(a_m=a)-\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{low}}\mathrm{I}(a_m=a)}{\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{high}}f_{m,a}-\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{low}}f_{m,a}}. \tag{3.5}$$

By Hoeffding's inequality and the union bound, for any $\delta > 0$, with probability $1-\delta$,

$$\begin{aligned}|\alpha-\hat{\alpha}| &\leq \frac{\sqrt{\ln(4/\delta)M}}{\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{high}}f_{m,a}-\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{low}}f_{m,a}}\\ &= (1/Z_a)\sqrt{\ln(4/\delta)/M}\;,\end{aligned} \tag{3.6}$$

where

$$Z_a=\frac{1}{M/2}\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{high}}f_{m,a}-\frac{1}{M/2}\sum_{m:\mathcal{I}_m\in\mathcal{M}_a^{low}}f_{m,a}\;.$$

The quantity Z_a measures the difference between the mean value of $f_{m,a}$ among instances with

"high" values of $f_{m,a}$ and the mean value of $f_{m,a}$ among instances with "low" values. While this quantity is data-dependent, standard uniform convergence theory tells us that it is stable once the data set is large. From Equation 3.6, we know that if there is an action a for which this difference is sufficiently high, then it is possible to obtain an accurate estimate of α given enough data. If, on the other hand, no such a exists, it follows that there is very little variance in the population distribution over the sample. We argue below that it is not necessary to learn α in order to mimic the behavior of an agent i if this is the case.

For now, assume that Z_a is sufficiently large for at least one value of a, and call this value a^*. We can use the estimate of α to obtain estimates of the weights for each action. From Equation 3.4, it is clear that for any a,

$$w_a = \frac{\mathrm{E}\left[\sum_{m:\mathcal{I}_m \in \mathcal{M}} \mathrm{I}(a_m = a)\right] - \alpha \sum_{m:\mathcal{I}_m \in \mathcal{M}} f_{m,a}}{(1-\alpha)M} \; .$$

We estimate this weight using

$$\hat{w}_a = \frac{\sum_{m:\mathcal{I}_m \in \mathcal{M}} \mathrm{I}(a_m = a) - \hat{\alpha} \sum_{m:\mathcal{I}_m \in \mathcal{M}} f_{m,a}}{(1-\hat{\alpha})M} \; . \tag{3.7}$$

The following lemma shows that given sufficient data, the error in these estimates is small when Z_{a^*} is large.

Lemma 8 *Let $a^* = \mathrm{argmax}_{a \in \mathcal{S}} Z_a$, and let $\hat{\alpha}$ be calculated as in Equation 3.5 with $a = a^*$. For each $a \in \mathcal{S}$, let $\hat{w}_a$ be calculated as in Equation 3.7. For sufficiently large M, for any $\delta > 0$, with probability $1 - \delta$,*

$$|\alpha - \hat{\alpha}| \leq (1/Z_{a^*}) \sqrt{\ln((4+2K)/\delta)/M} \; ,$$

and for all actions a,

$$|w_a - \hat{w}_a| \leq \frac{((1-\hat{\alpha})Z_{a^*}/\sqrt{2} + 2)\sqrt{\ln((4+2K)/\delta)}}{Z_{a^*}(1-\hat{\alpha})^2\sqrt{M} - (1-\hat{\alpha})\sqrt{\ln((4+2K)/\delta)}} \; .$$

The proof of this lemma appears in Appendix A3.2. It relies heavily on the following technical lemma for bounding the error of estimated ratios, which is proved in Appendix A3.3 and used

frequently throughout the remainder of this chapter.

Lemma 9 *For any positive u, $\hat{u}$, v, $\hat{v}$, k, and ϵ such that $\epsilon k < v$, if $|u - \hat{u}| \leq \epsilon$ and $|v - \hat{v}| \leq k\epsilon$, then*

$$\left| \frac{u}{v} - \frac{\hat{u}}{\hat{v}} \right| \leq \frac{\epsilon(v + uk)}{v(v - \epsilon k)} .$$

Now that we have bounds on the error of the estimated parameters, we can bound the expected $\mathcal{L}_1$ distance between the estimated model and the real model.

Lemma 10 *For sufficiently large M,*

$$\begin{aligned} & \mathrm{E}_{\mathbf{f} \sim D^f} \left[\sum_{a \in \mathcal{S}} |(\alpha f_a + (1-\alpha) w_a) - (\hat{\alpha} f_a + (1-\hat{\alpha}) \hat{w}_a)| \right] \\ \leq \quad & \frac{2\sqrt{\ln((4+2K)/\delta)}}{Z_{a^*}\sqrt{M}} + \min \left\{ \frac{K(Z_{a^*}/\sqrt{2}+2)\sqrt{\ln((4+2K)/\delta)}}{Z_{a^*}(1-\hat{\alpha})\sqrt{M} - \sqrt{\ln((4+2K)/\delta)}}, 2(1-\hat{\alpha}) \right\} . \end{aligned}$$

In this proof of this lemma, which appears in Appendix A3.4, the quantity

$$\sum_{a \in \mathcal{S}} |(\alpha f_a + (1-\alpha) w_a) - (\hat{\alpha} f_a + (1-\hat{\alpha}) \hat{w}_a)|$$

is bounded *uniformly* for all $\mathbf{f}$ using the error bounds. The bound on the expectation follows immediately.

It remains to show that we can still bound the error when Z_{a^*} is zero or very close to zero. We present a light sketch of the argument here; more details appear in Appendix A3.5.

Let η_a and μ_a be the true median and mean of the distribution from which the random variables $f_{m,a}$ are drawn. Let f_a^{high} be the mean value of the distribution over $f_{m,a}$ *conditioned on* $f_{m,a} > \eta_a$. Let $\bar{f}_a^{high}$ be the empirical average of $f_{m,a}$ conditioned on $f_{m,a} > \eta_a$. Finally, let $\hat{f}_a^{high} = (2/M) \sum_{m:\mathcal{I}_m \in \mathcal{M}_a^{high}} f_{m,a}$ be the empirical average of $f_{m,a}$ conditioned on $f_{m,a}$ being greater than the *empirical* median. We can calculate $\hat{f}_a^{high}$ from data.

We can apply standard arguments from uniform convergence theory to show that f_a^{high} is close to $\bar{f}_a^{high}$, and in turn that $\bar{f}_a^{high}$ is close to $\hat{f}_a^{high}$. Similar statements can be made for the analogous quantities f_a^{low}, $\bar{f}_a^{low}$, and $\hat{f}_a^{low}$. By noting that $Z_a = \hat{f}_a^{high} - \hat{f}_a^{low}$ this implies that if Z_a is small,

then the probability that a random value of $f_{m,a}$ is far from the mean μ_a is small. When this is the case, it is not necessary to estimate α directly. Instead, we set $\hat{\alpha} = 0$ and

$$\hat{w_a} = \frac{1}{M} \sum_{m:\mathcal{I}_m \in \mathcal{M}} I(a_m = a) \ .$$

Applying Hoeffding's inequality again, it is easy to show that for each a, $\hat{w_a}$ is very close to $\alpha\mu_a + (1-\alpha)w_a$, and from here it can be argued that the $\mathcal{L}_1$ distance between the estimated model and the real model is small.

Thus for any distribution D over state vectors, regardless of the corresponding value of Z_{a^*}, it is possible to build an accurate model for the strategy of agent i in polynomial time. By Theorem 8, this implies that the class is polynomially learnable from collective behavior.

Theorem 9 *The class of crowd affinity mixture model strategies is polynomially learnable from collective behavior.*

3.5.2 Learning Crowd Affinity Multiplicative Models

In Section 3.3.2, we introduced the crowd affinity multiplicative model. In this model, strategies are parameterized only by a weight vector $\mathbf{w}$. The probability that agent i chooses action a is simply $f_a w_a / \sum_{b \in \mathcal{S}} f_b w_b$.

Although the motivation for this model is similar to that for the mixture model, the dynamics of the system are quite different (see the simulations and discussion in Section 3.3), and a very different algorithm is necessary to learn individual strategies. In this section, we show that this class is polynomially learnable from collective behavior, and sketch the corresponding learning algorithm. The algorithm we present is based on a simple but powerful observation. In particular, consider the following random variable:

$$\chi_a^m = \begin{cases} 1/f_{m,a} & \text{if } f_{m,a} > 0 \text{ and } a_m = a \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

Suppose that for all m such that $\mathcal{I}_m \in \mathcal{M}$, it is the case that $f_{m,a} > 0$. Then by the definition of

the strategy class and linearity of expectation,

$$\begin{aligned} \mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}} \chi_a^m\right] &= \sum_{m:\mathcal{I}_m\in\mathcal{M}} \frac{1}{f_{m,a}}\left(\frac{f_{m,a}w_a}{\sum_{s\in\mathcal{S}} f_{m,s}w_s}\right) \\ &= w_a \sum_{m:\mathcal{I}_m\in\mathcal{M}} \frac{1}{\sum_{s\in\mathcal{S}} f_{m,s}w_s}\,, \end{aligned}$$

where the expectation is over the randomness in the agent's strategy. Notice that this expression is the product of two terms. The first, w_a, is precisely the value we would like to calculate. The second term is something that depends on the set of instances $\mathcal{M}$, but *does not* depend on action a. This leads to the key observation at the core of our algorithm. Specifically, if we have a second action b such that $f_{m,b} > 0$ for all m such that $\mathcal{I}_m \in \mathcal{M}$, then

$$\frac{w_a}{w_b} = \frac{\mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}} \chi_a^m\right]}{\mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}} \chi_b^m\right]}\,.$$

Although we do not know the values of these expectations, we can approximate them arbitrarily well given enough data. Since we have assumed (so far) that $f_{m,a} > 0$ for all $m \in \mathcal{M}$, and we know that $f_{m,a}$ represents a fraction of the population, it must be the case that $f_{m,a} \geq 1/N$ and $\chi_a^m \in [0, N]$ for all m. By a standard application of Hoeffding's inequality and the union bound, we see that for any $\delta > 0$, with probability $1 - \delta$,

$$\left|\sum_{m:\mathcal{I}_m\in\mathcal{M}} \chi_a^m - \mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}} \chi_a^m\right]\right| \leq \sqrt{\frac{N\ln(2/\delta)}{2|\mathcal{M}|}}\,. \tag{3.8}$$

This leads almost immediately to the following lemma. The role of β in this lemma may appear somewhat mysterious. It comes the fact that we are bounding the error of a ratio of two terms. An application of Lemma 9 using the bound in Equation 3.8 gives us a factor of $\chi_{a,b} + \chi_{b,a}$ in the numerator and a factor of $\chi_{b,a}$ in the denominator. This is problematic only when $\chi_{a,b}$ is significantly larger than $\chi_{b,a}$.

Lemma 11 *Suppose that $f_{m,a} > 0$ and $f_{m,b} > 0$ for all m such that $\mathcal{I}_m \in \mathcal{M}$. Then for any $\delta > 0$, with probability $1 - \delta$, for any $\beta > 0$, if $\chi_{a,b} \leq \beta\chi_{b,a}$ and $\chi_{b,a} \geq 1$, then if $|\mathcal{M}| \geq N\ln(2/\delta)/2$,*

then

$$\left|\frac{w_a}{w_b} - \frac{\sum_{m:\mathcal{I}_m \in \mathcal{M}} \chi_a^m}{\sum_{m:\mathcal{I}_m \in \mathcal{M}} \chi_b^m}\right| \leq \frac{(1+\beta)\sqrt{N \ln(2/\delta)}}{\sqrt{2|\mathcal{M}|} - \sqrt{N \ln(2/\delta)}} .$$

If we are fortunate enough to have a sufficient number of data instances for which $f_{m,a} > 0$ for all $a \in \mathcal{S}$, then this lemma supplies us with a way of approximating the ratios between all pairs of weights and subsequently approximating the weights themselves. In general, however, this may not be the case. Luckily, it is possible to estimate the ratio of the weights of each pair of actions a and b that are used together frequently by the population using only those data instances in which at least one agent is choosing each. Formally, define

$$\mathcal{M}_{a,b} = \{\mathcal{I}_m \in \mathcal{M} : f_{m,a} > 0, f_{m,b} > 0\} .$$

Lemma 11 tells us that if $\mathcal{M}_{a,b}$ is sufficiently large, and there is at least one instance $\mathcal{I}_m \in \mathcal{M}_{a,b}$ for which $a_m = b$, then we can approximate the ratio between w_a and w_b well.

What if one of these assumptions does not hold? If we are not able to collect sufficiently many instances in which $f_{m,a} > 0$ and $f_{m,b} > 0$, then standard uniform convergence results can be used to show that it is very unlikely that we see a new instance for which $f_a > 0$ and $f_b > 0$. This idea is formalized in the following lemma, the proof of which is in Appendix A3.6.

Lemma 12 *For any $M < |\mathcal{M}|$, for any $\delta \in (0,1)$, with probability $1-\delta$,*

$$\Pr_{\mathbf{f}\sim D^f}\left[\exists a, b \in \mathcal{S} : f_a > 0, f_b > 0, |\mathcal{M}_{a,b}| < M\right] \leq \frac{K^2}{2}\left(\frac{M}{|\mathcal{M}|} + \sqrt{\frac{\ln(K^2/(2\delta))}{2|\mathcal{M}|}}\right) .$$

Similarly, if $\chi_{a,b} = \chi_{b,a} = 0$, then a standard uniform convergence argument can be used to show that it is unlikely that agent i would ever select action a or b when $f_{m,a} > 0$ and $f_{m,b} > 0$. We will see that in this case, it is not important to learn the ratio between these two weights.

Using these observations, we can accurately model the behavior of agent i. The model consists of two phases. First, as a preprocessing step, we calculate a quantity

$$\chi_{a,b} = \sum_{m:\mathcal{I}_m \in \mathcal{M}_{a,b}} \chi_a^m$$

for each pair $a, b \in \mathcal{S}$. Then, each time we are presented with a state $\mathbf{f}$, we calculate a set of

weights for all actions a with $f_a > 0$ on the fly.

For a fixed $\mathbf{f}$, let $\mathcal{S}'$ be the set of actions $a \in \mathcal{S}$ such that $f_a > 0$. By Lemma 12, if the data set is sufficiently large, then we know that with high probability, it is the case that for all $a, b \in \mathcal{S}'$, $|\mathcal{M}_{a,b}| \geq M$ for some threshold M.

Now, let $a^* = \text{argmax}_{a \in \mathcal{S}'} |\{b : b \in \mathcal{S}', \chi_{a,b} \geq \chi_{b,a}\}|$. Intuitively, if there is sufficient data, a^* should be the action in $\mathcal{S}'$ with the highest weight, or have a weight arbitrarily close to the highest. Thus for any $a \in \mathcal{S}'$, Lemma 11 can be used to bound our estimate of w_a/w_{a^*} with a value of β arbitrarily close to 1. Noting that

$$\frac{w_a}{\sum_{s \in \mathcal{S}'} w_s} = \frac{w_a/w_{a^*}}{\sum_{s \in \mathcal{S}'} w_s/w_{a^*}} ,$$

we approximate the *relative* weight of action $a \in \mathcal{S}'$ with respect to the other actions in $\mathcal{S}'$ using

$$\hat{w}_a = \frac{\chi_{a,a^*}/\chi_{a^*,a}}{\sum_{s \in \mathcal{S}'} \chi_{s,a*}/\chi_{a^*,s}} ,$$

and simply let $\hat{w}_a = 0$ for any $a \notin \mathcal{S}'$. Applying Lemma 9, we find that for all $a \in \mathcal{S}'$, with high probability,

$$\left| \frac{w_a}{\sum_{s \in \mathcal{S}'} w_s} - \hat{w}_a \right| \leq \frac{(1+\beta)K\sqrt{N \ln(2K^2/\delta)}}{\sqrt{2M} - (1+\beta)K\sqrt{N \ln(2K^2/\delta)}} ,$$

where M is the lower bound on $|\mathcal{M}_{a,b}|$ for all $a, b \in \mathcal{S}'$, and β is close to 1. With this bound in place, it is straightforward to show that we can apply Lemma 9 once more to bound the expected $\mathcal{L}_1$,

$$\text{E}_{\mathbf{f} \sim D^f} \left[\sum_{a \in \mathcal{S}} \left| \frac{w_a f_a}{\sum_{s \in \mathcal{S}} w_s f_s} - \frac{\hat{w}_a f_a}{\sum_{s \in \mathcal{S}} \hat{w}_s f_s} \right| \right] ,$$

and that the bound goes to 0 at a rate of $O(1/\sqrt{M})$ as the threshold M grows. More details are given in Appendix A3.7.

Since it is possible to build an accurate model of the strategy of agent i in polynomial time under any distribution D over state vectors, we can again apply Theorem 8 to see that this class is polynomially learnable from collective behavior.

Theorem 10 *The class of crowd affinity multiplicative model strategies is polynomially learnable from collective behavior.*

3.6 Open Questions

We have introduced a computational model for learning from collective behavior, and populated it with some initial general theory and algorithmic results for crowd affinity models. In addition to positive or negative results for further agent strategy classes, there are a number of other general directions of interest for future research. These include extensions of our model to agnostic [79] settings, in which we relax the assumption that every agent strategy falls in a known class, and to reinforcement learning [121] or multiagent learning [70, 115] settings, in which the learning algorithm may itself be a member of the population being modeled and wishes to learn an optimal policy with respect to some reward function.

It would also be enlightening to study what collective information can or cannot be learned in a setting in which the learner cannot see *which* agents choose each action at each time step, but only *how many* agents choose each. This would more accurately capture voting scenarios in which only anonymized polling data is available.

Chapter 4

The Trade-Offs of Learning from Expert Advice

Suppose that each day, we are interested in predicting whether average stock prices are more likely to rise or fall. We may choose to base our predictions on advice from friends, family, and coworkers, in addition to journalists, bloggers, or financial analysts. The extensive and still-growing literature on "no-regret" learning has established that on any sequence of trials in which the predictions of a set of individuals, referred to as *experts*, are observed, it is possible to maintain a dynamically weighted prediction whose average performance asymptotically approaches that of the best single expert *in hindsight* as the number of time steps grows. Surprisingly, such guarantees hold even in a fully adversarial setting, with no distributional assumptions on the experts' performance [25].

Despite the impressiveness of such guarantees, competing with the best single expert is not always good enough. This style of guarantee has teeth only when we make the implicit assumption that there exist a small number of individuals in the population who dramatically outperform the rest. The goal of the algorithm then boils down to the "needle in a haystack" idea of finding and tracking these superior experts. This goal is simply too weak if all experts in a population have similar performance. When this is the case, the aggressive style of updates required to make strong general guarantees can result in the algorithm performing poorly compared to even the worst expert in the population!

This chapter contains a study of no-regret learning in a bicriteria setting. We examine not

only the standard notion of regret to the best expert (formally defined in Section 4.2 below), but also the regret to the average of all experts, the regret to any fixed mixture of experts, and the regret to the worst expert. This study leads to a new understanding of the limitations of existing no-regret algorithms, as well as new algorithms with novel performance guarantees. Specifically, we show that *any* algorithm that achieves only $O(\sqrt{T})$ cumulative regret to the best expert on a sequence of T trials must, in the worst case, suffer regret $\Omega(\sqrt{T})$ to the average, and that for a wide class of update rules that includes many existing no-regret algorithms (such as Exponential Weights [98, 55] and Follow the Perturbed Leader [72]), the product of the regret to the best and the regret to the average is, in the worst case, $\Omega(T)$. We then describe and analyze two alternate new algorithms that both achieve cumulative regret only $O(\sqrt{T}\log T)$ to the best expert and have only *constant* regret to any given fixed distribution over experts (that is, with no dependence on either T or the number of experts N). The key to the first algorithm is the gradual increase in the "aggressiveness" of updates in response to observed divergences in expert performances. The second algorithm is a simple twist on standard exponential-update algorithms.

The material in this chapter is based on joint work with Eyal Even-Dar, Michael Kearns, and Yishay Mansour [52]. The *D-Prod* algorithm in Section 4.4.3 grew out of a series of discussions with Tong Zhang.

4.1 Overview

Beginning at least as early as the 1950s, the literature on no-regret learning has established the following type of result. Consider any sequence of T trials in which the predictions of N individuals or *experts* are observed. Suppose that on each trial, each expert receives a reward or *gain* based on the quality of his prediction. For example, an expert might receive a gain of 1 for each correct prediction he makes and a gain of 0 for each incorrect prediction. Consider an algorithm that maintains a dynamic set of weights over the experts, and define the gain of the algorithm on a given time step to be the weighted average of the expert's gains; this can be interpreted as the expected gain the algorithm would receive if it chose a single expert to follow on each time step according to its current distribution. There exist such algorithms whose cumulative regret to the best single expert *in hindsight* (that is, the difference between the cumulative gains of the best performing expert

after the full sequence has been revealed and the cumulative gains of the algorithm) is guaranteed to be $O(\sqrt{T \log N})$, with absolutely no statistical assumptions on the sequence. Such results are especially interesting in light of the fact that even in known *stochastic* models, there is a matching lower bound of $\Omega(\sqrt{T \log N})$. The term "no-regret" derives from the fact that the per-step regret is only $O(\sqrt{(\log N)/T})$, which approaches zero as T becomes large.

In this chapter, we revisit no-regret learning, but with a bicriteria performance measure that is of both practical and philosophical interest. More specifically, in addition to looking at the cumulative regret to the *best* expert in hindsight, we simultaneously analyze the regret to the *average* gain of all experts (or more generally, any fixed weighting of the experts). For comparisons to the average, the gold standard will be only *constant* regret (independent of T and N). This is a sensible goal since if we were to consider regret to the average in isolation, *zero* regret would easily achieved by simply leaving the weights uniform at all times.

Our results establish strict trade-offs between regret to the best expert and the regret to the average in this setting, demonstrate that most known algorithms manage this trade-off poorly, and provide new algorithms with near optimal bicriteria performance. On the practical side, our new algorithms augment traditional no-regret results with a "safety net": while still managing to track the best expert near-optimally, they are guaranteed to never underperform the average (or any other given fixed weighting of experts) by more than just constant regret. On the philosophical side, the bicriteria analyses and lower bounds shed new light on prior no-regret algorithms, showing that the unchecked aggressiveness of their updates can indeed cause them to badly underperform simple benchmarks like the average.

Viewed at a suitably high level, many existing no-regret algorithms have a similar flavor. These algorithms maintain a distribution over the experts that is adjusted according to performance. Since we would like to compete with the best expert, a "greedy" or "momentum" algorithm that rapidly adds weight to an outperforming expert (or set of experts) is natural. Most known algorithms shift weight between competing experts at a rate proportional to $1/\sqrt{T}$, in order to balance the tracking of the current best expert with the possibility of this expert's performance suddenly dropping. Updates on the scale of $1/\sqrt{T}$ can be viewed as "aggressive", at least in comparison to the minimal average update of $1/T$ required for any interesting learning effects. (If updates are $o(1/T)$, the algorithm cannot make even a constant change to any given weight in T steps.)

Algorithm:	*If Regret to Best Is:*	*Then Regret to Average Is:*
Any Algorithm	$O(\sqrt{T})$	$\Omega(\sqrt{T})$
Any Algorithm	$\leq \sqrt{T \log T}/10$	$\Omega(T^{\epsilon})$
Any Difference Algorithm	$O(T^{\frac{1}{2}+\alpha})$	$\Omega(T^{\frac{1}{2}-\alpha})$

Table 4.1: Summary of the lower bounds presented in this chapter.

How poorly can existing regret minimization algorithms perform with respect to the average? Consider a sequence of gains for two experts where the gains for expert 1 are $1, 0, 1, 0, \cdots$, while the gains for expert 2 are $0, 1, 0, 1, \cdots$. Typical regret minimization algorithms (such as Exponential Weights [98, 55], Follow the Perturbed Leader [72], and the Prod algorithm [26]) will yield a gain of $T/2 - \Theta(\sqrt{T})$, meeting their guarantee of $O(\sqrt{T})$ regret with respect to the best expert. However, this performance leaves something to be desired. Note that in this example the performance of the best expert, worst expert, and average of the experts is identically $T/2$. Thus all of the algorithms mentioned above actually suffer a regret to the average (and to the worst expert) of $\Omega(\sqrt{T})$. The problem stems from the fact that in all even time steps the probability of expert 1 is exactly $1/2$. After expert 1 observes a gain of 1 we increase its probability by $c/\sqrt{T}$, where the precise value of c depends on the specific algorithm. Therefore in odd steps the probability of expert 2 is only $(1/2 - c/\sqrt{T})$. Note that adding a third expert, which is defined as the average of the original two, would not change this.[1]

This work establishes a sequence of results that demonstrate the inherent tension between regret to the best expert and the average, illuminates the problems of existing algorithms in managing this tension, and provides new algorithms that enjoy optimal bicriteria performance guarantees.

On the negative side, we show that *any* algorithm that has a regret of $O(\sqrt{T})$ to the best expert must suffer a regret of $\Omega(\sqrt{T})$ to the average in the worst case. We also show that any regret minimization algorithm that achieves at most $\sqrt{T \log T}/10$ regret to the best expert, must, in the worst case, suffer regret $\Omega(T^{\epsilon})$ to the average, for some constant $\epsilon \geq 0.02$. These lower bounds are established even when $N = 2$. A summary of these bounds is presented in Table 4.1.

On the positive side, we describe a new algorithm, *PhasedAggression*, that almost matches the

[1]The third expert would clearly have a gain of $1/2$ at every time step. At odd time steps, the weight of the first expert would be $1/3 + c/\sqrt{T}$, while that of the second expert would be $1/3 - c/\sqrt{T}$, resulting in a regret of $\Omega(\sqrt{T})$ to the average.

Algorithm:	*Regret to Best:*	*Regret to Average:*	*Regret to Worst:*
PhasedAggression	$O(\sqrt{T \log N}(\log T + \log\log N))$	O(1)	O(1)
D-Prod	$O(\sqrt{T \log N} + \sqrt{T/\log N}\log T)$	O(1)	O(1)
BestWorst	$O(N\sqrt{T \log N})$	$O(\sqrt{T \log N})$	0
EW	$O(T^{\frac{1}{2}+\alpha} \log N)$	$O(T^{\frac{1}{2}-\alpha})$	$O(T^{\frac{1}{2}-\alpha})$

Table 4.2: Summary of the algorithmic results presented in this chapter.

lower bounds above. Given any algorithm whose cumulative regret to the best expert is at most R (which may be a function of T and N but not of any data-dependent measures), we can use it to derive an algorithm whose regret to the best expert is $O(R \log R)$ with only constant regret to the average (or any given fixed distribution over the experts). Using an $O(\sqrt{T \log N})$ regret algorithm, this gives regret to the best of $O(\sqrt{T \log N}(\log T + \log\log N))$. In addition, we show how to use an R-regret algorithm to derive an algorithm with regret $O(NR)$ to the best expert and *zero* regret to the worst expert. These algorithms treat the given R-regret algorithm as a black box.

PhasedAggression is somewhat different from many of the traditional regret minimization algorithms, especially in its use of *restarts* that are driven by observed differences in expert performance. (Restarts have been used previously in the literature, but for other purposes [25].) We show that this difference is no coincidence. For a wide class of update rules that includes many existing algorithms (such as Weighted Majority/Exponential Weights, Follow the Perturbed Leader, and Prod), we show that the product of the regret to the best and the regret to the average is $\Omega(T)$. This establishes a frontier from which such algorithms inherently cannot escape. Furthermore, any point on this frontier can in fact be achieved by such an algorithm (i.e., a standard multiplicative update rule with an appropriately tuned learning rate). However, we show it is possible to circumvent the lower bound by using an algorithm "similar to" Prod to achieve guarantees close to those achieved by *PhasedAggression* without the use of restarts. This algorithm, *D-Prod*, escapes the lower bound by using a modified update rule that directly depends on the average of the experts' instantaneous gains at each time step. Our algorithmic results are summarized in Table 4.2.

It is worth noting that it is not possible in general to guarantee $o(\sqrt{T})$ regret to any arbitrary *pair* of distributions, D_1 and D_2. Consider a setting in which there are only two experts. Suppose distribution D_1 places all weight on one expert, while distribution D_2 places all weight on a second.

Competing simultaneously with both distributions is then equivalent to competing with the best expert, so we cannot expect to do better than known lower bounds of $\Omega(\sqrt{T})$.

Previous work by Auer et al. [5] considered adapting the learning rate of expert algorithms gradually. However, the goal of their work was to get an any-time regret bound without using the standard doubling technique and thus it is not surprising that their algorithm performance under the bicriteria setting is similar to the other existing algorithms. Vovk [128] also considered trade-offs in best expert algorithms. His work examined for which values of a and b it is possible for an algorithm's gain to be bounded by $aG_{best,T} + b \log N$, where $G_{best,T}$ is the gain of the best expert.

Chapter Outline: In Section 4.2, we review the standard framework for learning with experts. In Section 4.3, we provide an analysis of the trade-off between regret to the best and average for the broad class of difference algorithms, showing that the product between the two regrets for this class is $\Theta(T)$. In Section 4.4, we go on to show how this frontier can be broken by non-difference algorithms that gradually increase the aggressiveness of their updates. We first show how a very simple algorithm can enjoy standard regret bounds compared to the best expert in terms of T (though worse in terms of N), while having *zero* cumulative regret to the worst, and then present two alternative algorithms that compete well with both the best expert and the average with only logarithmic dependence on N. A general lower bound that holds for *any* algorithm is given in Section 4.5, and we conclude with some open questions in Section 4.6.

4.2 The Experts Framework

We consider the classic experts framework, in which each expert $i \in \{1, \cdots, N\}$ receives a gain $g_{i,t} \in [0, 1]$ at each time step t.[2] The cumulative gain of expert i up to time t is $G_{i,t} = \sum_{t'=1}^{t} g_{i,t'}$. We denote the average cumulative gain of the experts by time t as $G_{avg,t} = (1/N) \sum_{i=1}^{N} G_{i,t}$, and the gain of the best and worst expert as $G_{best,t} = \max_i G_{i,t}$ and $G_{worst,t} = \min_i G_{i,t}$. For any fixed distribution D over the experts, we define the gain of this distribution to be $G_{D,t} = \sum_{i=1}^{N} D(i) G_{i,t}$.

At each time t, an algorithm $\mathcal{A}$ assigns a weight $w_{i,t}$ to each expert i. These weights are normalized to probabilities $p_{i,t} = w_{i,t}/W_t$ where $W_t = \sum_i w_{i,t}$. Algorithm $\mathcal{A}$ then receives a gain

[2] All results presented in this chapter can be generalized to hold for instantaneous gains in any bounded region.

$g_{\mathcal{A},t} = \sum_{i=1}^{N} p_{i,t} g_{i,t}$. The cumulative gain of algorithm $\mathcal{A}$ up to time t is $G_{\mathcal{A},t} = \sum_{t'=1}^{t} g_{\mathcal{A},t'} = \sum_{t'=1}^{t} \sum_{i=1}^{N} p_{i,t'} g_{i,t'}$.

The standard goal of an algorithm in this setting is to minimize the regret to the best expert at a fixed time T. In particular, we would like to minimize the regret $R_{best,\mathcal{A},T} = \max\{G_{best,T} - G_{\mathcal{A},T}, 1\}$. [3] In this work, we are simultaneously concerned with minimizing both this regret and the regret to the average and worst expert, $R_{avg,\mathcal{A},T} = \max\{G_{avg,T} - G_{\mathcal{A},T}, 1\}$ and $R_{worst,\mathcal{A},T} = \max\{G_{worst,T} - G_{\mathcal{A},T}, 1\}$ respectively, in addition to the regret $R_{D,\mathcal{A},T}$ to a given distribution D, which is defined similarly.

While in general the bounds on the regret of the algorithms can be defined using the time (for example, a regret of $O(\sqrt{T})$) and these bounds are tight, this is considered a crude estimate and better measures are at hand. We present our positive results in terms of the maximal absolute gains $G_{max} = \max_i G_{i,T}$.

4.3 The $\Theta(T)$ Frontier for Difference Algorithms

We begin our results with an analysis of the trade-off between regret to the best and average for a wide class of existing algorithms, showing that the product between the two regrets for this class is $\Theta(T)$. A more general lower bound that holds for *any* algorithm is provided in Section 4.5.

Definition 7 (Difference Algorithm) *We call an algorithm $\mathcal{A}$ a* difference algorithm *if, when $N = 2$ and instantaneous gains are restricted to $\{0, 1\}$, the normalized weights $\mathcal{A}$ places on each of the two experts depend only on the difference between the experts' cumulative gains. In other words, $\mathcal{A}$ is a difference algorithm if there exists a function f such that when $N = 2$ and $g_{i,t} \in \{0, 1\}$ for all i and t, $p_{1,t} = f(d_t)$ and $p_{2,t} = 1 - f(d_t)$ where $d_t = G_{1,t} - G_{2,t}$.*

This simple definition is sufficient for the purposes of stating our lower bound below. Exponential Weights [98, 55], Follow the Perturbed Leader [72], and the Prod algorithm [26] are all examples of difference algorithms. (For Prod, this follows from the restriction on the instantaneous gains to $\{0, 1\}$.)

[3] This minimal value of 1 makes the presentation of the trade-off "nicer" (for example in the statement of Theorem 11), but has no real significance otherwise.

4.3.1 The Difference Frontier Lower Bound

Theorem 11 *Let $\mathcal{A}$ be any difference algorithm. Then there exists a sequence of expert gains of length T such that $R_{best,\mathcal{A},T} \cdot R_{avg,\mathcal{A},T} \geq R_{best,\mathcal{A},T} \cdot R_{worst,\mathcal{A},T} = \Omega(T)$.*

Proof: For simplicity, assume that T is an even integer. We will consider the behavior of the difference algorithm $\mathcal{A}$ on two sequences of expert payoffs. Both sequences involve only two experts with instantaneous gains in $\{0, 1\}$. (Since the theorem provides a lower bound, it is sufficient to consider an example in this restricted setting.) Assume without loss of generality that initially $p_{1,1} \leq 1/2$.

In the first sequence, S_1, Expert 1 has a gain of 1 at every time step while Expert 2 always has a gain 0. Let ρ be the first time t at which $\mathcal{A}$ has $p_{1,t} \geq 2/3$. If such a ρ does not exist, then $R_{best,\mathcal{A},T} = \Omega(T)$ and we are done. Assuming such a ρ does exist, $\mathcal{A}$ must have regret $R_{best,\mathcal{A},T} \geq \rho/3$ since it loses at least $1/3$ to the best expert on each of the first ρ time steps and cannot compensate for this later.

Since the probability of Expert 1 increases from $p_{1,1} \leq 1/2$ to at least $2/3$ in ρ time steps in S_1, there must be one time step $\tau \in [2, \rho]$ in which the probability of Expert 1 increased by at least $1/(6\rho)$, i.e., $p_{1,\tau} - p_{1,\tau-1} \geq 1/(6\rho)$. The second sequence S_2 we consider is as follows. For the first τ time steps, Expert 1 will have a gain of 1 (as in S_1). For the last τ time steps, Expert 1 will have a gain of 0. For the remaining $T - 2\tau$ time steps (in the range $[\tau, T - \tau]$), the gain of Expert 1 will alternate $0, 1, 0, 1, \cdots$. Throughout the sequence, Expert 2 will have a gain of 1 whenever Expert 1 has a gain of 0 and a gain of 0 every time Expert 1 has a gain of 1. This implies that each expert has a gain of exactly $T/2$ (and hence $G_{best,T} = G_{avg,T} = G_{worst,T} = T/2$).

During the period $[\tau, T - \tau]$, consider a pair of consecutive times such that $g_{1,t} = 0$ and $g_{1,t+1} = 1$. Since $\mathcal{A}$ is a difference algorithm we have that $p_{1,t} = p_{1,\tau}$ and $p_{1,t+1} = p_{1,\tau-1}$. The gain of algorithm $\mathcal{A}$ in time steps t and $t + 1$ is $(1 - p_{1,\tau}) + p_{1,\tau-1} \leq 1 - 1/(6\rho)$, since $p_{1,\tau} - p_{1,\tau-1} \geq 1/(6\rho)$. In every pair of time steps t and $T - t$, for $t \leq \tau$, the gain of $\mathcal{A}$ in those times steps is exactly 1, since the difference between the experts is identical at times t and $T - t$, and hence the probabilities are identical. This implies that the total gain of the algorithm $\mathcal{A}$ is at most

$$\tau + \frac{T - 2\tau}{2}\left(1 - \frac{1}{6\rho}\right) \leq \frac{T}{2} - \frac{T - 2\tau}{12\rho}\,.$$

On sequence S_1, the regret of algorithm $\mathcal{A}$ with respect to the best expert is $\Omega(\rho)$. Therefore, if $\rho \geq T/4$ we are done. Otherwise, on sequence S_2, the regret with respect to the average and worst is $\Omega(T/\rho)$. The theorem follows. ■

4.3.2 A Difference Algorithm Achieving the Frontier

We now show that the standard Exponential Weights (EW) algorithm with an appropriate choice of the learning rate parameter η [55] is a difference algorithm achieving the trade-off described in Section 4.3.1, thus rendering it tight for this class. Recall that for all experts i, EW assigns initial weights $w_{i,1} = 1$, and at each subsequent time t, updates weights with $w_{i,t+1} = e^{\eta G_{i,t}} = w_{i,t}e^{\eta g_{i,t}}$. The probability with which expert i is chosen at time t is then given by $p_{i,t} = w_{i,t}/W_t$ where $W_t = \sum_{j=1}^{N} w_{j,t}$.

Theorem 12 *Let $G^* \leq T$ be an upper bound on G_{max}. For any α such that $0 \leq \alpha \leq 1/2$, let $EW = EW(\eta)$ with $\eta = (G^*)^{-(1/2+\alpha)}$. Then $R_{best,EW,T} \leq (G^*)^{1/2+\alpha}(1 + \ln N)$ and $R_{avg,EW,T} \leq (G^*)^{1/2-\alpha}$.*

Proof: These bounds can be derived using a series of bounds on the quantity $\ln(W_{T+1}/W_1)$. First we bound this quantity in terms of the gain of the best expert and the gain of EW. This piece of the analysis is standard (see, for example, Theorem 2.4 in [25]). The first piece of the bound follows from the fact that $W_1 = N$ and that $w_{best,t+1} \leq W_{t+1}$.

$$\eta G_{best,T} - \ln N \leq \ln\left(\frac{W_{T+1}}{W_1}\right) .$$

The second pieces follows from a simple application of Taylor approximation.

$$\ln\left(\frac{W_{T+1}}{W_1}\right) \leq \left(\eta + \eta^2\right) G_{EW,T} . \tag{4.1}$$

Therefore, we derive that $G_{best,T} - G_{EW,T} \leq \eta G_{EW,T} + \ln N/\eta$.

Next we bound the same quantity in terms of the average cumulative gain, using the fact that the arithmetic mean of a set of nonnegative numbers is always greater than or equal to the geometric

mean.

$$\ln\left(\frac{W_{T+1}}{W_1}\right) = \ln\left(\frac{\sum_{i=1}^{N} w_{i,T+1}}{N}\right) \geq \ln\left(\left(\prod_{i=1}^{N} w_{i,T+1}\right)^{\frac{1}{N}}\right) \quad (4.2)$$
$$= \frac{1}{N}\sum_{i=1}^{N} \ln w_{i,T+1} = \frac{1}{N}\sum_{i=1}^{N} \eta G_{i,T} = \eta G_{avg,T} \ .$$

Combined with the upper bound in Equation 4.1, this gives us $G_{avg,T} - G_{EW,T} \leq \eta G_{EW,T}$.

Note that if $G_{best,T} \leq G_{EW,T}$, both the regret to the best expert and the regret to the average will be minimal, so we can assume this is not the case and replace the term $G_{EW,T}$ on the right hand side of these bounds with $G_{best,T}$ which is in turn bounded by G^*. This yields the pair of bounds $G_{best,T} - G_{EW,T} \leq \eta G^* + \ln N/\eta$ and $G_{avg,T} - G_{EW,T} \leq \eta G^*$.

By changing the value of η, we can construct different trade-offs between the two bounds. Setting $\eta = (G^*)^{-(1/2+\alpha)}$ yields the desired result. ■

This trade-off can be generalized to hold when we would like to compete with an arbitrary distribution D by initializing $w_{i,1} = D(i)$ and substituting an alternate inequality into (4.2). The $\ln(N)$ term in the regret to the best expert will be replaced by $max_{i \in N} \ln(1/D(i))$, making this practical only for distributions that lie inside the probability simplex and not too close to the boundaries.

4.4 Breaking the Difference Frontier

The results so far have established a $\Theta(T)$ frontier on the product of regrets to the best and average experts for difference algorithms. In this section, we will show how this frontier can be broken by non-difference algorithms that gradually increase the aggressiveness of their updates via a series of restarts invoked by observed differences in performance so far. As a warm-up, we first show how a very simple algorithm that is not a difference algorithm can enjoy standard regret bounds compared to the best expert in terms of T (though worse in terms of N), while having *zero* cumulative regret to the worst. In Sections 4.4.2 and 4.4.3, we present two alternative algorithms that compete well with both the best expert and the average with only logarithmic dependence on N.

4.4.1 Regret to the Best and Worst Experts

Using a standard regret-minimization algorithm as a black box, we can produce a very simple algorithm, *BestWorst*, that achieves a clear trade-off between regret to the best expert and regret to the worst expert. Let $\mathcal{A}$ be a regret minimization algorithm such that $R_{best,\mathcal{A},T} \leq R$ for some R which may be a function of T and N but not of any data dependent measures. We define the modified algorithm *BestWorst($\mathcal{A}$,R)* as follows. While the difference between the cumulative gains of the best and worst experts is smaller than NR, *BestWorst($\mathcal{A}$,R)* places equal weight on each expert, playing the average. After the first time τ at which this condition is violated, it begins running a fresh instance of algorithm $\mathcal{A}$ and continues to use $\mathcal{A}$ until the end of the sequence. See Figure 4.1 below.

Figure 4.1 The *BestWorst* algorithm for N experts.

```
// Input:  An algorithm A and value R such that we are
// guaranteed R_{best,A,T} ≤ R

while (G_{best,t} − G_{worst,t} ≤ NR) do
   Use probabilities p_{i,t} = 1/N for all i
end while
Reset and run algorithm A for all remaining time steps.
```

Until time τ, this algorithm must be performing at least as well as the worst expert since it is playing the average. At time τ, the algorithm's gain must be R more than that of the worst expert since the gain of the best expert is NR above the gain of the worst. Now since from time τ algorithm $\mathcal{A}$ is run, we know that the gain of *BestWorst($\mathcal{A}$,R)* in the final $T - \tau$ time steps will be no more than R less than the gain of any other expert. Therefore, *BestWorst($\mathcal{A}$,R)* will maintain a lead over the expert that was worst in the first phase (and thus also the worst expert overall). In addition, the regret of the algorithm to the best expert will be bounded by NR, since up to time τ it will have a regret of at most $(N-1)R$ with respect to the best expert. This establishes the following theorem.

Theorem 13 *Let $\mathcal{A}$ be a regret minimization algorithm with regret at most R to the best expert and let BW be* BestWorst($\mathcal{A}$,R)*. Then*

$$R_{best,BW,T} \leq NR$$

and

$$G_{BW,T} \geq G_{worst,T} \ .$$

It follows immediately that using a standard regret minimization algorithm with $R = O(\sqrt{T \log N})$ as the black box, we can achieve a regret of $O(N\sqrt{T \log N})$ to the best expert while maintaining a lead over the worst.

4.4.2 PhasedAggression

Again using any standard regret-minimization algorithm as a black box, we can produce an algorithm, *PhasedAggression*, that achieves a trade-off between regret to the best expert and regret to the average without sacrificing too much in terms of the dependence on N. Figure 4.2 shows *PhasedAggression*. This algorithm can achieve a constant regret to *any* specified distribution D, not only the average, with no change to the bounds. The name of the algorithm refers to the fact that it operates in distinct phases separated by restarts, with each phase more aggressive than the last.

Figure 4.2 The *PhasedAggression* algorithm for N experts.

```
// Input:  An algorithm A and value R such that we are
// guaranteed R_{best,A,T} ≤ R, and a distribution D with
// which we would like to compete

for k = 1 to ⌊log(R)⌋ do
   Let η = 2^{k-1}/R
   Reset and run a new instance of A
   while (G^p_{best,t} − G^p_{D,t} < 2R) do
      Feed A the previous gains g_{t-1} and let q_t be its distribution
      Use p_t = ηq_t + (1 − η)D
   end while
end for
Reset and run a new instance of A until time T
```

The idea behind the algorithm is rather simple. We take a regret minimization algorithm $\mathcal{A}$, and mix between $\mathcal{A}$ and the target distribution D. As the gain of the best expert exceeds the gain of D by larger amounts, we put more and more weight on the regret minimization algorithm $\mathcal{A}$, "resetting" $\mathcal{A}$ to its initial state at the start of each phase. Once the weight on $\mathcal{A}$ has been increased, it is never decreased again. In other words, in each successive phase of this algorithm

(or reduction), weight is moved from something that is not learning at all (the fixed distribution D) to an algorithm that is implicitly learning aggressively (the given algorithm $\mathcal{A}$). New phases are invoked only in response to greater and greater outperformance by the current best expert, allowing the amount of aggression to increase only as needed.

Theorem 14 *Let $\mathcal{A}$ be any algorithm with regret R to the best expert, D be any distribution, and PA be an instantiation of* PhasedAggression*$(\mathcal{A}, R, D)$. Then $R_{best,PA,T} \leq 2R(\log R + 1)$ and $R_{D,PA,T} \leq 1$.*

Proof: We will again analyze the performance of the algorithm compared to the best expert and the distribution D both during and at the end of any phase k. First consider any time t during phase k. The regret of the algorithm is split between the regret of the fixed mixture and the regret of the no-regret algorithm according to their weights. Since $\mathcal{A}$ is an R-regret algorithm its regret to both the best expert and to the distribution D is bounded by R, and thus the regret of the algorithm due to the weight on $\mathcal{A}$ is $2^{k-1}/R$ times R. With the remaining $1 - (2^{k-1}/R)$ weight, the regret to the best expert is bounded by $2R$ since $G^p_{best,t} - G^p_{D,t} < 2R$ during the phase, and its regret to distribution D is 0. Thus at any time t during phase k we have

$$G^p_{best,t} - G^p_{PA,t} \leq R\left(\frac{2^{k-1}}{R}\right) + 2R\left(1 - \frac{2^{k-1}}{R}\right) \leq 2R$$

and

$$G^p_{D,t} - G^p_{PA,t} \leq R\left(\frac{2^{k-1}}{R}\right) = 2^{k-1}\ .$$

Now consider what happens when the algorithm exits phase k. A phase is only exited at some time t such that $G^p_{best,t} - G^p_{D,t} \geq 2R$. Since $\mathcal{A}$ is R-regret, its gain (in the current phase) will be within R of the gain of the best expert, resulting in the algorithm PA gaining a *lead* over distribution D for the phase: $G^p_{PA,t} - G^p_{D,t} \geq R(2^{k-1}/R) = 2^{k-1}$.

Combining these inequalities, it is clear that if the algorithm ends in phase k at time T, then

$$G_{best,T} - G_{PA,T} \leq 2Rk \leq 2R(\log R + 1)$$

and

$$G_{D,T} - G_{PA,T} \leq 2^{k-1} - \sum_{j=1}^{k-1} 2^{j-1} = 2^{k-1} - (2^{k-1} - 1) = 1 \; .$$

These inequalities hold even when the algorithm reaches the final phase and has all of its weight on $\mathcal{A}$, thus proving the theorem. ■

4.4.3 D-Prod

While Exponential Weights and Prod are both difference algorithms and cannot avoid the $\Omega(T)$ frontier lower bound, it is possible to create an algorithm with exponential updates that can achieve guarantees similar to those of *PhasedAggression* without requiring the use of restarts. This can be accomplished via a simple algorithm that we refer to as *D-Prod* since its update rules and analysis are inspired by those of Prod.

D-Prod differs from Prod in two important ways that together allow it to compete well with the best expert while simultaneously guaranteeing only constant regret to a given fixed distribution D. First, an additional expert (denoted expert 0) representing the distribution D is added with a large prior weight. Second, the update rule is modified to take into account the *difference* in performance between each expert and the distribution D rather than the gains of each expert alone. Because of this second modification, *D-Prod* is *not* a difference algorithm and is able to avoid the $\Omega(T)$ frontier.

Formally, let $g_{i,t} \in [0,1]$ be the gain of expert i at time t as before for $i \in \{1, \ldots, N\}$, and let $g_{0,t}$ be the instantaneous gain of the distribution D (or the special expert 0). Each expert i starts with an initial or prior weight $w_{i,1} = \mu_i$. At each time step, weights are updated using $w_{i,t+1} = w_{i,t}(1 + \eta(g_{i,t} - g_{0,t}))$.

Lemma 13 *For any expert i (including the special expert 0), for any $\eta \leq 1/2$,*

$$G_{D-Prod,T} \geq G_{i,T} + \frac{\ln(\mu_i)}{\eta} - \eta \sum_{t=1}^{T} (g_{i,t} - g_{0,t})^2 \; .$$

The proof is nearly identical to the proof of Lemma 2 of Cesa-Bianchi et al. [26]. Notice that when $i = 0$ (the special expert), the last term in the bound is 0. The following theorem shows how

to set the parameters η and μ to achieve a constant error rate to the distribution D without losing much with respect to the best expert.

Theorem 15 *Let $\eta = \sqrt{\ln N/T}$, $\mu_0 = 1 - \eta$, and $\mu_i = \eta/N$ for $i \in \{1, \ldots, N\}$. Then*

$$R_{best,D-Prod,T} = O\left(\sqrt{T \ln N} + \sqrt{\frac{T}{\ln N}} \ln T\right)$$

and $R_{D,D-Prod,T} = O(1)$.

4.5 Sketch of A General Lower Bound

So far we have seen that a wide class of existing algorithms (namely all difference algorithms) is burdened with a stark best/average regret trade-off, but that this frontier can be avoided by simple algorithms that tune how aggressively they update, in phases modulated by the observed payoffs so far. What is the limit of what can be achieved in our bicriteria regret setting?

In this section we give a pair of general lower bounds that hold for *all* algorithms. The bounds are stated for the average but once again hold for any fixed distribution D. These lower bounds come close to the upper bound achieved by the algorithms described in the previous section. An outline of the proof is sketched below.

Theorem 16 *Any algorithm with regret $O(\sqrt{T})$ to the best expert must have regret $\Omega(\sqrt{T})$ to the average in the worst case. Furthermore, any algorithm with regret at most $\sqrt{T \log T}/10$ to the best expert must have regret $\Omega(T^\epsilon)$ to the average in the worst case for some positive constant $\epsilon \geq 0.02$.*

More specifically, we show that for any constant $\alpha > 0$, there exists a constant $\beta > 0$ such that for sufficiently large values of T (i.e. $T > (150\alpha)^2$), for any algorithm $\mathcal{A}$, there exists a sequence of gains $\mathbf{g}$ of length T such that if $R_{best,\mathcal{A},T} \leq \alpha\sqrt{T}$ then $R_{avg,\mathcal{A},T} \geq \beta\sqrt{T}$ even when $N = 2$. Additionally, for any constant $\alpha' \leq 1/10$ there exist constants $\beta' > 0$ and $\epsilon > 0$ such that for sufficiently large values of T (i.e., $T > 2^{(10\alpha')^2}$), for any algorithm $\mathcal{A}$, there exists a sequence of gains of length T such that if $R_{best,\mathcal{A},T} \leq \alpha'\sqrt{T \log T}$ then $R_{avg,\mathcal{A},T} \geq \beta' T^\epsilon$.

The proof of this theorem begins by defining a procedure for creating a "bad" sequence of gains $\mathbf{g}$, specifically designed to fool the algorithm $\mathcal{A}$. Whenever the algorithm updates its weights

aggressively, the procedure assigns positive gains to the second-place expert, inducing mean reversion. On the contrary, when the algorithm updates its weights conservatively, the procedure assigns positive gains to the best expert, causing added momentum. The sequence can then be divided into a number of (possibly noncontiguous) segments based on the algorithm's weights at each time. By first analyzing the maximum amount that the algorithm can gain over the average and the minimum amount it can lose to the average in each segment, and then bounding the total number of segments possible under the assumption that an algorithm is no-regret, we can show that it is not possible for an algorithm to have $O(\sqrt{T})$ regret to the best expert without having $\Omega(\sqrt{T})$ regret to the average. More details of the proof are given in Appendix A4.1.

4.6 Open Questions

The results in this chapter depend heavily on the fact that the gain of the algorithm at each time step is the weighted average of the gains of the experts. This can be interpreted as the expected gain that the algorithm would receive if it chose a single expert to follow on each time step according to its current distribution and subsequently received the gain of this expert. We might instead consider a scenario in which the algorithm is able to combine the advice of the experts in more sophisticated ways and receive a gain based on this combination, for example based on the squared loss of the combined prediction. It is not clear if similar results could be proved in such a setting. It would also be interesting to determine whether or not similar trade-offs exist in the related portfolio setting [37, 68].

We currently do not know whether or not it is possible to strengthen Theorem 16 to say that any algorithm with regret $O(\sqrt{T \log T})$ to the best expert must have regret $\Omega(T^{\epsilon})$ to the average for some constant $\epsilon > 0$. Such a result would further close the gap between our positive and negative results, but would potentially require a different style of proof.

Chapter 5

Aggregating Opinions Via Prediction Markets and Machine Learning

Suppose we are interested in obtaining an estimate of the probability that the average global temperature will rise over the next five years. One way of forming an estimate is to invite people to bet on it. Prediction markets are financial markets designed to aggregate individual beliefs about the outcome of an event into a single prediction. In practice, these predictions are often highly accurate, frequently outperforming forecasts made by domain experts.

Prediction markets generally operate in isolation, and over relatively small outcome spaces. For example, a typical horse race market might allow bettors to choose one of n horses to win, even though the true outcome space of the event is much larger. (There are $n!$ possible permutations of horses in the race.) This is due in large part to the intensive amount of computation required to store and update an exponential number of linked prices. Chen et al. [29] show that when the market is operated by a central auctioneer who simply performs riskless order matching, the auctioneer's matching problem can be made tractable by enforcing appropriate restrictions on the betting language used (in particular, by allowing only bets of the form "either horse A or horse C will finish in third place"), even though the outcome space is exponentially large. However, such markets suffer from low liquidity. It may be the case that no matches are made even when there are parties willing to bet.

In this chapter, we investigate the computational complexity of market maker pricing algorithms for combinatorial prediction markets. We focus on Hanson's popular logarithmic market scoring rule market maker (LMSR) [62, 63], which is *always* willing to accept bets on *any* outcome at some price. Our goal is to implicitly maintain correct LMSR prices across an exponentially large outcome space. We examine both permutation combinatorics, where outcomes are permutations of objects, and Boolean combinatorics, where outcomes are combinations of binary events. We look at three restrictive languages that limit what traders can bet on. Even with severely limited languages, we find that LMSR pricing is $\#\mathsf{P}$-hard, even when the same language admits polynomial-time matching without the market maker.

On the positive side, we point out and explore the previously unnoticed connection between LMSR prices and the weights used in online learning with experts. Using this connection, we propose an approximation technique for pricing permutation markets based on an algorithm for online permutation learning.

The contents of this chapter are based on joint work with Yiling Chen, Lance Fortnow, Nicolas Lambert, and David Pennock [31], with helpful suggestions from Sampath Kannan.

5.1 Overview

One way to elicit information is to ask people to bet on it. A *prediction market* is a common forum where people bet with each other or with a market maker [105]. They are commonly known by a variety of names, including information markets, securities markets, event markets, event futures, and idea futures [105]. A typical binary prediction market allows bets along one dimension, for example, either for or against Barack Obama to win a second term as US President in 2012. In this case, bettors would trade shares of securities that pay off $\$1$ if and only if Obama wins. If the current market price of a share is $\$p$, then a rational, risk-neutral bettor should be willing to buy shares if he believes the true probability of Obama winning is greater than p. Conversely, he should be willing to sell shares at this price if he believes that the true probability of Obama winning is lower.

The current price per share thus provides an estimate of the population's collective belief about how likely it is that Obama will win a second term. In fact, under certain assumptions (in particular,

the assumption that all bettors are risk-neutral Bayesians with a common prior), it is possible to prove that the trading price converges to what is known as a *rational expectations equilibrium* that reflects the probability estimate that one would obtain by combining the private side information of each member of the population [103, 59]. Indeed, in practice forecasts obtained through prediction markets are frequently more accurate than forecasts provided by experts in a broad spectrum of settings. The price of orange juice futures is a better predictor of weather than the National Weather Service forecasts [109], Oscar markets are more accurate at predicting winners than expert columnists [106], and election markets are more accurate than national polls [17]; see Pennock and Sami [105] or Ledyard et al. [91] for a range of other examples.

Thousands of one- or small-dimensional markets exist today, each operating independently. At the racetrack, betting on a horse to win does not directly impact the odds for that horse to finish among the top two, as logically it should, because the two bet types are handled separately. On the contrary, a *combinatorial prediction market* is a central clearinghouse for handling logically-related bets defined on a combinatorial space. For example, the outcome space might be all $n!$ possible permutations of n horses in a horse race, while bets are properties of permutations such as "horse A finishes third" or "horse A beats horse B." Alternately, the outcome space might be all 2^{50} possible state-by-state results for the 2012 US Presidential election, while bets are Boolean statements such as "the Republican candidate wins in Florida but not in Pennsylvania or Ohio."

Chen et al. [29] show that when the market is operated by a central auctioneer performing risk-less order matching, the matching problem can be solved efficiently in some special cases, even when the outcome space is exponentially large. However, low liquidity marginalizes the value of these prediction markets, and combinatorics only exacerbates the problem by dividing traders' attention among an exponential number of outcomes. A combinatorial matching market—the combinatorial generalization of a standard double auction—may simply fail to find any trades [53, 29]. In contrast, an *automated market maker* is always willing to trade on *every* bet at some price. A combinatorial market maker implicitly or explicitly maintains prices across all (exponentially many) outcomes, thus allowing any trader at any time to place any bet, if transacted at the market maker's quoted price.

Hanson's logarithmic market scoring rule market maker (LMSR) [62, 63], which is described in Section 5.2, is becoming the de facto standard market maker for prediction markets, largely

because it has a number of desirable properties, including bounded loss that grows logarithmically in the number of outcomes, infinite liquidity, and modularity that respects some independence relationships. LMSR is used by a number of companies, including Microsoft, inklingmarkets.com, thewsx.com, and yoonew.com, and is the subject of a number of research studies [30, 91, 28].

In this chapter, we analyze the computational complexity of LMSR in several combinatorial betting scenarios. We examine both permutation combinatorics and Boolean combinatorics. We show that both computing instantaneous prices and computing payments of transactions are #P-hard in all cases we examine, even when we restrict participants to very simplistic and limited types of bets. For example, in the horse race analogy, if participants can place bets only of the form "horse A finishes in position N", then pricing these bets properly according to LMSR is #P-hard, even though matching up bets of the exact same form (with no market maker) can be done in polynomial time [29].

On a more positive note, we examine an approximation algorithm for LMSR pricing in permutation markets that makes use of powerful techniques from the literature on online learning with expert advice [25, 98, 56]. We point out and examine the striking parallels that exist between the specific form of standard LMSR prices and the expert weights employed by the Weighted Majority algorithm [98]. We then show how a recent extension of Weighted Majority to permutation learning [67] can be transformed into an approximation algorithm for pricing in permutation markets in which the market maker is guaranteed to have bounded loss.

Fortnow et al. [53] study the computational complexity of finding acceptable trades among a set of bids in a Boolean combinatorial market. In their setting, the center is an *auctioneer* who takes no risk, only matching together willing traders. They study a call market setting in which bids are collected together and processed once en masse, and show that the auctioneer matching problem is co-NP-complete when orders are divisible and Σ_2^p-complete when orders are indivisible, but identify some tractable special cases. As mentioned above, Chen et al. [29] analyze the the auctioneer matching problem for betting on permutations, examining subset betting and pair betting. They give a polynomial-time algorithm for matching divisible subset bets, but show that matching pair bets is NP-hard.

The work closest to our own is that of Chen et al. [32], who study a special case of Boolean combinatorics in which participants bet on how far a team will advance in a single elimination

tournament, for example a sports playoff like the NCAA college basketball tournament. They provide a polynomial-time algorithm for LMSR pricing in this setting based on a Bayesian network representation of prices. They also show that LMSR pricing is NP-hard for a very general bidding language.

Chapter Outline: In the next section, we describe Hanson's logarithmic market scoring rule market maker in detail, including the calculation of prices. Section 5.3 contains a quick review of some known results from complexity theory that are referred to throughout the remainder of the chapter. Section 5.4 and 5.5 contain our hardness results for permutation betting and Boolean betting respectively. Finally, in Section 5.6, we discuss the connection between calculating LMSR prices and calculating expert weights in online learning, and show how the PermELearn algorithm of Helmbold and Warmuth [67] can be used to efficiently approximate prices for subset betting in permutation markets. We briefly discuss some open directions of research in Section 5.7.

5.2 Logarithmic Market Scoring Rules

Proposed by Hanson [62, 63], a logarithmic market scoring rule is an automated market maker mechanism that always maintains a consistent probability distribution over an outcome space Ω reflecting the market's estimate of the likelihood of each outcome. A generic LMSR offers a security corresponding to each possible outcome ω. The security associated to outcome ω pays off \$1 if the outcome ω happens, and \$0 otherwise. Let $\mathbf{q} = (q_\omega)_{\omega \in \Omega}$ indicate the number of outstanding shares for all securities. The LMSR market maker starts the market with some initial shares of securities, $\mathbf{q^0}$, which may be $\mathbf{0}$. The market keeps track of the outstanding shares of securities $\mathbf{q}$ at all times, and maintains a cost function

$$C(\mathbf{q}) = b \log \sum_{\omega \in \Omega} \mathrm{e}^{q_\omega / b}, \tag{5.1}$$

and an instantaneous price function for each security

$$p_\omega(\mathbf{q}) = \frac{\mathrm{e}^{q_\omega / b}}{\sum_{\tau \in \Omega} \mathrm{e}^{q_\tau / b}}, \tag{5.2}$$

where b is a positive parameter related to the depth of the market. The cost function captures the total money wagered in the market, and $C(\mathbf{q^0})$ reflects the market maker's maximum subsidy to the market. The instantaneous price function $p_\omega(\mathbf{q})$ gives the current cost per share of an infinitely small quantity of the security for outcome ω, and is the partial derivative of the cost function, i.e. $p_\omega(\mathbf{q}) = \partial C(\mathbf{q})/\partial q_\omega$. We use $\mathbf{p} = (p_\omega(\mathbf{q}))_{\omega\in\Omega}$ to denote the price vector. Traders buy and sell securities through the market maker. If a trader wishes to change the number of outstanding shares from $\mathbf{q}$ to $\tilde{\mathbf{q}}$, the cost of the transaction that the trader pays is $C(\tilde{\mathbf{q}}) - C(\mathbf{q})$, which equals the integral of the price functions following any path from $\mathbf{q}$ to $\tilde{\mathbf{q}}$.

When the outcome space is large, it is often natural to offer only compound securities on sets of outcomes. A compound security S pays \$1 if one of the outcomes in the set $S \subset \Omega$ occurs and \$0 otherwise. Such a security is the combination of all securities $\omega \in S$. Buying or selling q shares of the compound security S is equivalent to buying or selling q shares of each security $\omega \in S$. Let Θ denote the set of all allowable compound securities. Denote the outstanding shares of all compound securities as $Q = (q_S)_{S\in\Theta}$. The cost function can be written as

$$C(Q) = b \log \sum_{\omega\in\Omega} \mathrm{e}^{\sum_{S\in\Theta:\omega\in S} q_S/b} = b \log \sum_{\omega\in\Omega} \prod_{S\in\Theta:\omega\in S} \mathrm{e}^{q_S/b} \,. \tag{5.3}$$

The instantaneous price of a compound security S is computed as the sum of the instantaneous prices of the securities that compose the compound security S,

$$p_S(Q) = \frac{\sum_{\omega\in S} \mathrm{e}^{q_\omega/b}}{\sum_{\tau\in\Omega} \mathrm{e}^{q_\tau/b}} = \frac{\sum_{\omega\in S} \mathrm{e}^{\sum_{S'\in\Theta:\omega\in S'} q_{S'}/b}}{\sum_{\tau\in\Omega} \mathrm{e}^{\sum_{S'\in\Theta:\tau\in S'} q_{S'}/b}} = \frac{\sum_{\omega\in S} \prod_{S'\in\Theta:\omega\in S'} \mathrm{e}^{q_{S'}/b}}{\sum_{\tau\in\Omega} \prod_{S'\in\Theta:\tau\in S'} \mathrm{e}^{q_{S'}/b}} \,. \tag{5.4}$$

Logarithmic market scoring rules are so named because they are based on *logarithmic scoring rules*. A logarithmic scoring rule is a set of reward functions

$$\{s_\omega(\mathbf{r}) = a_\omega + b \log(r_\omega) : \omega \in \Omega\},$$

where $\mathbf{r} = (r_\omega)_{\omega\in\Omega}$ is a probability distribution over Ω, and a_ω is a free parameter. An agent who reports $\mathbf{r}$ is rewarded $s_\omega(\mathbf{r})$ if outcome ω happens. Logarithmic scoring rules are *proper* in the sense that when facing them a risk-neutral agent will truthfully report his subjective probability distribution to maximize his expected reward. A LMSR market can be viewed as a sequential

version of logarithmic scoring rule, because by changing market prices from $\mathbf{p}$ to $\tilde{\mathbf{p}}$ a trader's net profit is $s_\omega(\tilde{\mathbf{p}}) - s_\omega(\mathbf{p})$ when outcome ω happens. At any time, a trader in a LMSR market is essentially facing a logarithmic scoring rule.

LMSR markets have many desirable properties. They offer consistent pricing for combinatorial events. As market maker mechanisms, they provide infinite liquidity by allowing trades at any time. Although the market maker subsidizes the market, he is guaranteed a worst-case loss no greater than $C(\mathbf{q^0})$, which is $b \log n$ if $|\Omega| = n$ and the market starts with 0 share of every security. In addition, it is a dominant strategy for a myopic risk-neutral trader to reveal his probability distribution truthfully since he faces a proper scoring rule. Even for forward-looking traders, truthful reporting is an equilibrium strategy when traders' private information is independent conditional on the true outcome [30].

5.3 Complexity of Counting

We now briefly review some standard ideas and results from complexity theory that are used throughout the remainder of the chapter.

The well-known class NP contains questions that ask whether a search problem has a solution, such as whether a graph is 3-colorable. The class #P consists of functions that *count* the number of solutions of NP search questions, such as the number of 3-colorings of a graph.

A function g is #P-hard if, for every function f in #P, it is possible to compute f in polynomial time given an oracle for g. Clearly oracle access to such a function g could additionally be used to solve any NP problem, but in fact one can solve much harder problems too. Toda [122] showed that every language in the polynomial-time hierarchy can be solved efficiently with access to a #P-hard function.

To show a function g is a #P-hard function, it is sufficient to show that a function f reduces to g where f was previously known to be #P-hard. In this work, we use the following #P-hard functions to reduce from:

- **Permanent**: The permanent of an n-by-n matrix $A = (a_{i,j})$ is defined as

$$\mathrm{perm}(A) = \sum_{\sigma \in \Omega} \prod_{i=1}^{n} a_{i,\sigma(i)} \ , \tag{5.5}$$

where Ω is the set of all permutations over $\{1, 2, ..., n\}$. Computing the permanent of a matrix A containing only 0-1 entries is $\#\mathsf{P}$-hard [123].

- $\#$**2-SAT**: Counting the number of satisfying assignments of a formula given in conjunctive normal form with each clause having two literals is $\#\mathsf{P}$-hard [124].

- **Counting Linear Extensions**: Counting the number of total orders that extend a partial order given by a directed graph is $\#\mathsf{P}$-hard [22].

$\#\mathsf{P}$-hardness is the best we can achieve since all the functions in this chapter can themselves be reduced to some other $\#\mathsf{P}$ function.

5.4 LMSR for Permutation Betting

In this section we consider a particular type of market combinatorics in which the final outcome is a ranking over n competing candidates. Let the set of candidates be $\mathcal{N}_n = \{1, \ldots, n\}$, which is also used to represent the set of positions. In the setting, Ω is the set of all permutations over $\mathcal{N}_n$. An outcome $\sigma \in \Omega$ is interpreted as the scenario in which each candidate i ends up in position $\sigma(i)$. Chen et al. [29] propose two betting languages, *subset betting* and *pair betting*, for this type of combinatorics and analyze the complexity of the auctioneer's order matching problem for each. In what follows we address the complexity of operating an LMSR market for both betting languages.

5.4.1 Subset Betting

As in Chen et al. [29], participants in a LMSR market for subset betting may trade two types of compound securities: (1) a security of the form $\langle i|\Phi\rangle$ where $\Phi \subset \mathcal{N}_n$ is a subset of positions; and (2) a security $\langle \Psi|j\rangle$ where $\Psi \subset \mathcal{N}_n$ is a subset of candidates. The security $\langle i|\Phi\rangle$ pays off \$1 if candidate i stands at a position that is an element of Φ and \$0 otherwise. Similarly, the security $\langle \Psi|j\rangle$ pays off \$1 if any of the candidates in Ψ finishes at position j and \$0 otherwise. For example, in a horse race, participants can trade securities of the form "horse A will come in the second, fourth, or fifth place," or "either horse B or horse C will come in the third place."

Note that owning one share of $\langle i|\Phi\rangle$ is equivalent to owning one share of $\langle i|j\rangle$ for every $j \in \Phi$, and similarly owning one share of $\langle \Psi|j\rangle$ is equivalent to owing one share of $\langle i|j\rangle$ for every $i \in \Psi$.

We therefore restrict our attention to a simplified market where securities traded are of the form $\langle i|j\rangle$. We show that even in this simplified market it is #P-hard for the market maker to provide the instantaneous security prices, evaluate the cost function, or calculate payments for transactions, which implies that the running an LMSR market for the more general case of subset betting is also #P-hard.

Traders can trade securities $\langle i|j\rangle$ for all $i \in \mathcal{N}_n$ and $j \in \mathcal{N}_n$ with the market maker. Let $q_{i,j}$ be the total number of outstanding shares for security $\langle i|j\rangle$ in the market. Let $Q = (q_{i,j})_{i\in\mathcal{N}_n, j\in\mathcal{N}_n}$ denote the outstanding shares for all securities. The market maker keeps track of Q at all times. From Equation 5.4, the instantaneous price of security $\langle i|j\rangle$ is

$$p_{i,j}(Q) = \frac{\sum_{\sigma\in\Omega:\sigma(i)=j} \prod_{k=1}^{n} \mathrm{e}^{q_{k,\sigma(k)}/b}}{\sum_{\tau\in\Omega} \prod_{k=1}^{n} \mathrm{e}^{q_{k,\tau(k)}/b}} \,, \tag{5.6}$$

and from Equation 5.3, the cost function for subset betting is

$$C(Q) = b \log \sum_{\sigma\in\Omega} \prod_{k=1}^{n} \mathrm{e}^{q_{k,\sigma(k)}/b} \,. \tag{5.7}$$

We will show that computing instantaneous prices, the cost function, and/or payments of transactions for a subset betting market is #P-hard by a reduction from the problem of computing the permanent of a (0,1)-matrix.

Theorem 17 *It is #P-hard to compute instantaneous prices in a LMSR market for subset betting. Additionally, it is #P-hard to compute the value of the cost function.*

Proof: We show that if we could compute the instantaneous prices or the value of the cost function for subset betting for any quantities of shares purchased, then we could compute the permanent of any $(0, 1)$-matrix in polynomial time.

Let $A = (a_{i,j})$ be any n-by-n (0,1)-matrix, and define $N = n! + 1$. Note that $\prod_{i=1}^{n} a_{i,\sigma(i)}$ is either 0 or 1. From Equation 5.5, $\mathrm{perm}(A) \leq n!$ and hence $\mathrm{perm}(A) \mod N = \mathrm{perm}(A)$. We show how to compute $\mathrm{perm}(A) \mod N$ from prices in subset betting markets over up to n candidates in which for each pair of candidates i and j, $q_{i,j}$ shares of $\langle i|j\rangle$ have been purchased,

with

$$q_{i,j} = \begin{cases} b \ln N & \text{if } a_{i,j} = 0, \\ b \ln(N+1) & \text{if } a_{i,j} = 1. \end{cases} \tag{5.8}$$

Let $B = (b_{i,j})$ be a n-by-n matrix containing entries of the form $b_{i,j} = \mathrm{e}^{q_{i,j}/b}$. Note that $b_{i,j} = N$ if $a_{i,j} = 0$ and $b_{i,j} = N+1$ if $a_{i,j} = 1$. Thus, $\mathrm{perm}(A) \mod N = \mathrm{perm}(B) \mod N$. Thus, from Equation 5.6, the price for $\langle i|j \rangle$ in the market is

$$p_{i,j}(Q) = \frac{\sum_{\sigma \in \Omega : \sigma(i) = j} \prod_{k=1}^{n} b_{k,\sigma(k)}}{\sum_{\tau \in \Omega} \prod_{k=1}^{n} b_{k,\tau(k)}} = \frac{b_{i,j} \sum_{\sigma \in \Omega : \sigma(i) = j} \prod_{k \neq i} b_{k,\sigma(k)}}{\sum_{\tau \in \Omega} \prod_{k=1}^{n} b_{k,\tau(k)}} = \frac{b_{i,j} \cdot \mathrm{perm}(M_{i,j})}{\mathrm{perm}(B)}$$

where $M_{i,j}$ is the matrix obtained from B by removing the ith row and jth column. Thus the ability to efficiently compute prices gives us the ability to efficiently compute $\mathrm{perm}(M_{i,j})/\mathrm{perm}(B)$.

It remains to show that we can use this ability to compute $\mathrm{perm}(B)$. We do so by telescoping a sequence of prices. Let B_i be the matrix B with the first i rows and columns removed. From above, we have $\mathrm{perm}(B_1)/\mathrm{perm}(B) = p_{1,1}(Q)/b_{1,1}$. Define Q_m to be the $(n-m)$-by-$(n-m)$ matrix $(q_{i,j})_{i>m, j>m}$, that is, the matrix of quantities of securities $(q_{i,j})$ with the first k rows and columns removed. In a market with only $n-m$ candidates, applying the same technique to the matrix Q_m, we can obtain $\mathrm{perm}(B_{m+1})/\mathrm{perm}(B_m)$ from market prices for $m = 1, ..., (n-2)$. Thus by computing $n-1$ prices, we can compute

$$\left(\frac{\mathrm{perm}(B_1)}{\mathrm{perm}(B)}\right)\left(\frac{\mathrm{perm}(B_2)}{\mathrm{perm}(B_1)}\right)\cdots\left(\frac{\mathrm{perm}(B_{n-1})}{\mathrm{perm}(B_{n-2})}\right) = \left(\frac{\mathrm{perm}(B_{n-1})}{\mathrm{perm}(B)}\right).$$

Since B_{n-1} only has one element, we thus can compute $\mathrm{perm}(B)$ from market prices. Consequently, $\mathrm{perm}(B) \mod N$ gives $\mathrm{perm}(A)$.

Therefore, given a n-by-n $(0,1)$-matrix A, we can compute the permanent of A in polynomial time using prices in $n-1$ subset betting markets wherein an appropriate quantity of securities have been purchased.

Additionally, note that

$$C(Q) = b \log \sum_{\sigma \in \Omega} \prod_{k=1}^{n} b_{k,\sigma(k)} = b \log \mathrm{perm}(B) \, .$$

Thus if we can compute $C(Q)$, we can also compute $\mathrm{perm}(A)$.

As computing the permanent of a $(0,1)$-matrix is #P-hard, both computing market prices and computing the cost function in a subset betting market are #P-hard. ■

Corollary 1 *Computing the payment of a transaction in a LMSR for subset betting is #P-hard.*

Proof: Suppose the market maker starts the market with 0 share of every security. Denote Q^0 as the initial quantities of all securities. If the market maker can compute $C(\tilde{Q}) - C(Q)$ for any quantities $\tilde{Q}$ and Q, it can compute $C(Q) - C(Q^0)$ for any Q. As $C(Q^0) = b \log n!$, the market maker is able to compute $C(Q)$. By Theorem 17, computing the payment of a transaction is #P-hard. ■

5.4.2 Pair Betting

In contrast to subset betting, where traders bet on absolute positions for a candidate, pair betting allows traders to bet on the relative position of a candidate with respect to another. More specifically, traders buy and sell securities of the form $\langle i > j \rangle$, where i and j are candidates. The security pays off \$1 if candidate i ranks higher than candidate j (i.e., $\sigma(i) < \sigma(j)$ where σ is the final ranking of candidates) and \$0 otherwise. For example, traders may bet on events of the form "horse A beats horse B", or "candidate C receives more votes than candidate D".

As for subset betting, the current state of the market is determined by the total number of outstanding shares for all securities. Let $q_{i,j}$ denote the number of outstanding shares for $\langle i > j \rangle$. Applying Equations 5.3 and 5.4 to the present context, we find that the instantaneous price of the security $\langle i, j \rangle$ is given by

$$p_{i,j}(Q) = \frac{\sum_{\sigma \in \Omega : \sigma(i) < \sigma(j)} \prod_{i',j' : \sigma(i') < \sigma(j')} \mathrm{e}^{q_{i',j'}/b}}{\sum_{\tau \in \Omega} \prod_{i',j' : \tau(i') < \tau(j')} \mathrm{e}^{q_{i',j'}/b}} , \tag{5.9}$$

and the cost function for pair betting is

$$C(Q) = b \log \sum_{\sigma \in \Omega} \prod_{i,j : \sigma(i) < \sigma(j)} \mathrm{e}^{q_{i,j}/b} . \tag{5.10}$$

We show that computing prices, the value of the cost function, and/or payments of transactions for pair betting is #P-hard via a reduction from the problem of computing the number of linear

extensions to any partial ordering. The proof is in Appendix A5.1. The corollary then follows from a similar argument to the proof of Corollary 1.

Theorem 18 *It is #P-hard to compute instantaneous prices in a LMSR market for pair betting. Additionally, it is #P-hard to compute the value of the cost function.*

Corollary 2 *Computing the payment of a transaction in a LMSR for pair betting is #P-hard.*

5.5 LMSR for Boolean Betting

We now examine an alternate type of market combinatorics in which the final outcome is a conjunction of event outcomes. Formally, let $\mathcal{A}$ be event space, consisting of N individual events $A_1, \cdots, A_N$, which may or may not be mutually independent. We define the state space Ω to be the set of all possible joint outcomes for the N events, so that its size is $|\Omega| = 2^N$. A Boolean betting market allows traders to bet on Boolean formulas of these events and their negations. A security $\langle \phi \rangle$ pays off \$1 if the Boolean formula ϕ is satisfied by the final outcome and \$0 otherwise. For example, a security $\langle A_1 \vee A_2 \rangle$ pays off \$1 if and only if at least one of events A_1 and A_2 occurs, while a security $\langle A_1 \wedge A_3 \wedge \neg A_5 \rangle$ pays off \$1 if and only if the events A_1 and A_3 both occur and the event A_5 does not. Following the notational conventions of Fortnow et al. [53], we use $\omega \in \phi$ to mean that the outcome ω satisfies the Boolean formula ϕ. Similarly, $\omega \notin \phi$ implies that the outcome ω does not satisfy ϕ.

In this section, we focus our attention to LMSR markets for a very simple Boolean betting language, Boolean formulas of two events. We show that even when bets are only allowed to be placed on disjunctions or conjunctions of two events, it is still #P-hard to calculate the prices, the value of the cost function, and payments of transactions in a Boolean betting market operated by a LMSR market maker.

Let $\mathcal{X}$ be the set containing all elements of $\mathcal{A}$ and their negations. In other words, each event outcome $X_i \in \mathcal{X}$ is either A_j or $\neg A_j$ for some $A_j \in \mathcal{A}$. We begin by considering the scenario in which traders may only trade securities $\langle X_i \vee X_j \rangle$ corresponding to disjunctions of any two event outcomes.

Let $q_{i,j}$ be the total number of shares purchased by all traders for the security $\langle X_i \vee X_j \rangle$, which pays off \$1 in the event of any outcome ω such that $\omega \in (X_i \vee X_j)$ and \$0 otherwise. From

Equation 5.4, we can calculate the instantaneous price for the security $\langle X_i \vee X_j \rangle$ for any two event outcomes $X_i, X_j \in \mathcal{X}$ as

$$p_{i,j}(Q) = \frac{\sum_{\omega \in \Omega : \omega \in (X_i \vee X_j)} \prod_{1 \leq i' < j' \leq 2N : \omega \in (X_{i'} \vee X_{j'})} \mathrm{e}^{q_{i',j'}/b}}{\sum_{\tau \in \Omega} \prod_{1 \leq i' < j' \leq 2N : \tau \in (X_{i'} \vee X_{j'})} \mathrm{e}^{q_{i',j'}/b}}. \tag{5.11}$$

Note that if $X_i = \neg X_j$, $p_{i,j}(Q)$ is always \$1 regardless of how many shares of other securities have been purchased. According to Equation 5.3, the cost function is

$$C(Q) = b \log \sum_{\omega \in \Omega} \prod_{1 \leq i < j \leq 2N : \omega \in (X_i \vee X_j)} \mathrm{e}^{q_{i,j}/b}. \tag{5.12}$$

Theorem 19 shows that computing prices and the value of the cost function in such a market is #P-hard. The proof is via a reduction from the #2-SAT problem. The structure of the proof is extremely similar to that of the pair betting case, and is given in Appendix A5.2. The proof of the corollary is then nearly identical to the proof of Corollary 1.

Theorem 19 *It is #P-hard to compute instantaneous prices in a LMSR market for Boolean betting when bets are restricted to disjunctions of two event outcomes. Additionally, it is #P-hard to compute the value of the cost function in this setting.*

Corollary 3 *Computing the payment of a transaction in a LMSR for Boolean betting is #P-hard when traders can only bet on disjunctions of two events.*

If we impose that participants in a Boolean betting market may only trade securities corresponding to conjunctions of any two event outcomes, $\langle A_i \wedge A_j \rangle$, the following Corollary gives the corresponding complexity results.

Corollary 4 *It is #P-hard to compute instantaneous prices in a LMSR market for Boolean betting when bets are restricted to conjunctions of two event outcomes. Additionally, it is #P-hard to compute the value of the cost function in this setting, and #P-hard to compute the payment for a transaction.*

Proof: Buying q shares of security $\langle A_i \wedge A_j \rangle$ is equivalent to selling q shares of $\langle \neg A_i \vee \neg A_j \rangle$.

Thus if we can operate a Boolean betting market for securities of the type $\langle A_i \wedge A_j \rangle$ in polynomial time, we can also operate a Boolean betting market for securities of the type $\langle A_i \vee A_j \rangle$ in polynomial time. The result then follows from Theorem 19 and Corollary 3. ■

5.6 An Approximation Algorithm for Subset Betting

There is an interesting relationship between logarithmic market scoring rule market makers and a common class of algorithms for online learning in an experts setting. In this section, we elaborate on this connection, and show how results from the online learning community can be used to prove new results about an approximation algorithm for subset betting.

5.6.1 Review of the Experts Setting

We begin with a brief review the standard model of online learning with expert advice, which is described in more detail in Chapter 4. Recall that in Section 4.2, this model was defined in terms of expert *gains*. For the purposes of this chapter, it is easier to think in terms of *losses*, which are simply negated gains. We therefore start by reintroducing the setting using slightly altered notation.

At each time $t \in \{1, \cdots, T\}$, each expert $i \in \{1, \cdots, n\}$ receives a *loss* $\ell_{i,t} \in [0, 1]$. The *cumulative loss* of expert i at time T is $\mathcal{L}_{i,T} = \sum_{t=1}^{T} \ell_{i,t}$. No statistical assumptions are made about these losses, and in general, algorithms are expected to perform well even if the sequence of losses is chosen by an adversary.

An algorithm $\mathcal{A}$ maintains a current weight $w_{i,t}$ for each expert i, where $\sum_{i=1}^{n} w_{i,t} = 1$. These weights can be viewed as distributions over the experts. The algorithm then receives its own instantaneous loss $\ell_{\mathcal{A},t} = \sum_{i=1}^{n} w_{i,t}\ell_{i,t}$, which may be interpreted as the expected loss of the algorithm when choosing an expert according to the current distribution. The cumulative loss of $\mathcal{A}$ up to time T is then defined in the natural way as $\mathcal{L}_{\mathcal{A},T} = \sum_{t=1}^{T} \ell_{\mathcal{A},t} = \sum_{t=1}^{T} \sum_{i=1}^{n} w_{i,t}\ell_{i,t}$. A common goal in such online learning settings is to minimize an algorithm's *regret*. Here the regret is defined as the difference between the cumulative loss of the algorithm and the cumulative loss of an algorithm that would have "chosen" the best expert in hindsight by setting his weight to 1 throughout all the periods. Formally, the regret is given by $\mathcal{L}_{\mathcal{A},T} - \min_{i \in \{1,\cdots,n\}} \mathcal{L}_{i,T}$.

Many algorithms that have been analyzed in the online experts setting are based on exponential weight updates. As discussed in Chapter 4, these exponential updates allow the algorithm to quickly transfer weight to an expert that is outperforming the others. For example, in the Weighted Majority algorithm of Littlestone and Warmuth [98], the weight on each expert i is defined as

$$w_{i,t} = \frac{w_{i,t-1}\mathrm{e}^{-\eta\ell_{i,t}}}{\sum_{j=1}^{n} w_{j,t-1}\mathrm{e}^{-\eta\ell_{j,t}}} = \frac{\mathrm{e}^{-\eta\mathcal{L}_{i,t}}}{\sum_{j=1}^{n}\mathrm{e}^{-\eta\mathcal{L}_{j,t}}} \ , \tag{5.13}$$

where η is the *learning rate*, a small positive parameter that controls the magnitude of the updates. The following theorem gives a bound on the regret of Weighted Majority. For a proof of this result and a nice overview of learning with expert advice, see Cesa-Bianchi and Lugosi [25].

Theorem 20 (e.g., Cesa-Bianchi and Lugosi [25]) *Let $\mathcal{A}$ be the Weighted Majority algorithm with parameter η. After a sequence of T trials,*

$$\mathcal{L}_{\mathcal{A},T} - \min_{i\in\{1,\cdots,n\}} \mathcal{L}_{i,T} \leq \eta T + \frac{\ln(n)}{\eta} \ .$$

When T is known in advance, setting $\eta = \sqrt{\ln(n)/T}$ yields the standard $\sqrt{T\ln(n)}$ regret bound.

5.6.2 Relationship to LMSR Markets

There is a manifest similarity between the expert weights utilized by Weighted Majority and the prices in an LMSR market; simply compare the form of Equation 5.13 with the form of Equation 5.2. One might ask if the results from the experts setting can be applied to the analysis of prediction markets. Our answer is *yes*. For example, it is possible to use Theorem 20 to rediscover the well-known bound of $b\ln(n)$ for the loss of an LMSR market maker with n outcomes.

Let ϵ be a limit on the number of shares that a trader may purchase or sell at each time step; in other words, if a trader would like to purchase or sell q shares, this purchase must be broken down into $\lceil q/\epsilon \rceil$ separate purchases of ϵ or less shares. Note that the total number of time steps T needed to execute such a sequence of purchases and sales is proportional to $1/\epsilon$.

We will construct a sequence of loss functions in a setting with n experts to induce a sequence of weight matrices that correspond to the price matrices of the LMSR market. At each time step t,

let $p_{i,t} \in [0,1]$ be the instantaneous price of security i at the end of period t, and let $q_{i,t} \in [-\epsilon, \epsilon]$ be the number of shares of security i purchased during period t. Let $Q_{i,t}$ be the total number of shares of security i that have been purchased up to time t. Define the instantaneous loss of each expert as $\ell_{i,t} = (\epsilon - q_{i,t})/(\eta b)$. First notice that this loss is always in $[0,1]$ as long as $\eta \geq 2\epsilon/b$. From Equations 5.2 and 5.13, at each time t,

$$p_{i,t} = \frac{\mathrm{e}^{Q_{i,t}/b}}{\sum_{j=1}^{n} \mathrm{e}^{Q_{j,t}/b}} = \frac{\mathrm{e}^{\epsilon t/b - \eta \mathcal{L}_{i,t}}}{\sum_{j=1}^{n} \mathrm{e}^{\epsilon t/b - \eta \mathcal{L}_{j,t}}} = \frac{\mathrm{e}^{-\eta \mathcal{L}_{i,t}}}{\sum_{j=1}^{n} \mathrm{e}^{-\eta \mathcal{L}_{j,t}}} = w_{i,t} \ .$$

Applying Theorem 20, and rearranging terms, we find that

$$\max_{i \in \{1,\cdots,n\}} \sum_{t=1}^{T} q_{i,t} - \sum_{t=1}^{T} \sum_{i=1}^{n} p_{i,t} q_{i,t} \leq \eta^2 T b + b \ln(n).$$

The first term of the left-hand side is the maximum payment that the market maker needs to make, while the second terms of the left-hand side captures the total money the market maker has received. The right hand side is clearly minimized when η is set as small as possible. Setting $\eta = 2\epsilon/b$ (which, as we mentioned above, is the smallest value we can choose for η while guaranteeing that each expert's instantaneous loss is in $[0,1]$) gives us

$$\max_{i \in \{1,\cdots,n\}} \sum_{t=1}^{T} q_{i,t} - \sum_{t=1}^{T} \sum_{i=1}^{n} p_{i,t} q_{i,t} \leq 4\epsilon^2 T b + b \ln(n).$$

Since $T = O(1/\epsilon)$, the term $4\epsilon^2 T b$ goes to 0 as ϵ becomes very small. Thus in the limit as ϵ approaches 0, we get the well-known result that the worst-case loss of the market maker is bounded by $b \ln(n)$.

5.6.3 Considering Permutations

In recent work, Helmbold and Warmuth [67] show that many results from the standard experts setting can be extended to a setting in which, instead of competing with the best expert, the goal is to compete with the best permutation over n items. Here each permutation suffers a loss at each time step, and the goal of the algorithm is to maintain a weighting over permutations such that the cumulative regret to the best permutation is small. It is generally infeasible to treat each

permutation as an expert and run a standard algorithm since this would require updating $n!$ weights at each time step. Instead, they show that when the loss has a certain structure (in particular, when the loss of a permutation is the sum of the losses of each of the n mappings), an alternate algorithm can be used that requires tracking only n^2 weights in the form of an $n \times n$ doubly stochastic matrix.

Formally, let W^t be a doubly stochastic matrix of weights maintained by the algorithm $\mathcal{A}$ at time t. Here $W^t_{i,j}$ is the weight corresponding to the probability associated with item i being mapped into position j. Let $L^t \in [0,1]^{n \times n}$ be the loss matrix at time t. The instantaneous loss of a permutation σ at time t is $\ell_{\sigma,t} = \sum_{i=1}^{n} L^t_{i,\sigma(i)}$. The instantaneous loss of $\mathcal{A}$ is $\ell_{\mathcal{A},t} = \sum_{i=1}^{n} \sum_{j=1}^{n} W^t_{i,j} L^t_{i,j}$, the matrix dot product between W^t and L^t. Notice that $\ell_{\mathcal{A},t}$ is equivalent to the expectation over permutations σ drawn according to W^t of $\ell_{\sigma,t}$. The goal of the algorithm is to minimize the cumulative regret to the best permutation, $\mathcal{L}_{\mathcal{A},T} - \min_{\sigma \in \Omega} \mathcal{L}_{\sigma,T}$ where the cumulative loss is defined as before.

Helmbold and Warmuth go on to present an algorithm called PermELearn that updates the weight matrix in two steps. First, it creates a temporary matrix W', such that for every i and j, $W'_{i,j} = W^t_{i,j} \mathrm{e}^{-\eta L^t_{i,j}}$. It then obtains $W^{t+1}_{i,j}$ by repeatedly rescaling the rows and columns of W' until the matrix is doubly stochastic. Alternately rescaling rows and columns of a matrix M in this way is known as Sinkhorn balancing [116]. Normalizing the rows of a matrix is equivalent to pre-multiplying by a diagonal matrix, while normalizing the columns is equivalent to post-multiplying by a diagonal matrix. Sinkhorn [116] shows that this procedure converges to a unique doubly stochastic matrix of the form RMC where R and C are diagonal matrices if M is a positive matrix. Although there are cases in which Sinkhorn balancing does not converge in finite time, many results show that the number of Sinkhorn iterations needed to scale a matrix so that row and column sums are $1 \pm \epsilon$ is polynomial in $1/\epsilon$ [7, 73, 97].

The following theorem bounds the cumulative loss of the PermELearn in terms of the cumulative loss of the best permutation.

Theorem 21 (Helmbold and Warmuth [67]) *Let $\mathcal{A}$ be the PermELearn algorithm with parameter η. After a sequence of T trials,*

$$\mathcal{L}_{\mathcal{A},T} \leq \frac{n \ln(n) + \eta \min_{\sigma \in \Omega} \mathcal{L}_{\sigma,T}}{1 - \mathrm{e}^{-\eta}} \; .$$

5.6.4 Approximating Subset Betting

Using the PermELearn algorithm, it is simple to approximate prices for subset betting in polynomial time. We start with a $n \times n$ price matrix P^1 in which all entries are $1/n$. As before, traders may purchase securities of the form $\langle i|\Phi\rangle$ that pay off \$1 if and only if horse or candidate i finishes in a position $j \in \Phi$, or securities of the form $\langle \Psi|j\rangle$ that pay off \$1 if and only if a horse or candidate $i \in \Psi$ finishes in position j.

As in Section 5.6.2, each time a trader purchases or sells q shares, the purchase or sale is broken up into $\lceil q/\epsilon \rceil$ purchases or sales of ϵ shares or less, where $\epsilon > 0$ is a small constant.[1] Thus we can treat the sequence of purchases as a sequence of T purchases of ϵ or less shares, where $T = O(1/\epsilon)$. Let $q^t_{i,j}$ be the number of shares of securities $\langle i|\Phi\rangle$ with $j \in \Phi$ or $\langle \Psi|j\rangle$ with $i \in \Psi$ purchased at time t; then $q^t_{i,j} \in [-\epsilon, \epsilon]$ for all i and j.

The price matrix is updated in two steps. First, a temporary matrix P' is created where for every i and j, $P'_{i,j} = P^t_{i,j} \mathrm{e}^{q^t_{i,j}/b}$ where $b > 0$ is a parameter playing a similar role to b in Equation 5.2. Next, P' is Sinkhorn balanced to the desired precision, yielding an (approximately) doubly stochastic matrix P^{t+1}.

The following lemma shows that updating the price matrix in this way results in a price matrix that is equivalent to the weight matrix of PermELearn with particular loss functions.

Lemma 14 *The sequence of price matrices obtained by the approximation algorithm for subset betting on a sequence of purchases $q^t \in [-\epsilon, \epsilon]^{n \times n}$ is equivalent to the sequence of weight matrices obtained by running PermELearn(η) on a sequence of losses L^t with*

$$L^t_{i,j} = \frac{\epsilon - q^t_{i,j}}{\eta b}$$

for all i and j, for any $\eta \geq 2\epsilon/b$.

Proof: First note that for any $\eta \geq 2\epsilon/b$, $L^t_{i,j} \in [0, 1]$ for all t, i, and j, so the loss matrix is valid for PermELearn. P^1 and W^1 both contain all entries of $1/n$. Assume that $P^t = W^t$. When updating

[1] We remark that dividing purchases in this way has the negative effect of creating a polynomial time dependence on the quantity of shares purchased. However, this is not a problem if the quantity of shares bought or sold in each trade is bounded to start, which is a reasonable assumption. The additional time required is then linear only in $1/\epsilon$.

weights for time $t+1$, for all i and j,

$$P'_{i,j} = P^t_{i,j}\mathrm{e}^{q^t_{i,j}/b} = W^t_{i,j}\mathrm{e}^{q^t_{i,j}/b} = \mathrm{e}^{\epsilon/b}W^t_{i,j}\mathrm{e}^{-\epsilon/b+q^t_{i,j}/b} = \mathrm{e}^{\epsilon/b}W^t_{i,j}\mathrm{e}^{-\eta L^t_{i,j}} = \mathrm{e}^{\epsilon/b}W'_{i,j} \ .$$

Since the matrix W' is a constant multiple of P', the Sinkhorn balancing step will produce the same matrices. ■

Using this lemma, we can show that the difference between the amount of money that the market maker must distribute to traders in the worst case (i.e. when the true outcome is the outcome that pays off the most) and the amount of money collected by the market is bounded. We will see in the corollary below that as ϵ approaches 0, the worst case loss of the market maker approaches $bn\ln(n)$, regardless of the number of shares purchased. Unfortunately, if $\epsilon > 0$, this bound can grow arbitrarily large.

Theorem 22 *For any sequence of valid subset betting purchases q^t where $q^t_{i,j} \in [-\epsilon, \epsilon]$ for all t, i, and j, let $P^1, \cdots, P^T$ be the price matrices obtained by running the subset betting approximation algorithm. Then*

$$\max_{\sigma\in S_n}\sum_{t=1}^{T}\sum_{i=1}^{n} q^t_{i,\sigma(i)} - \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} P^t_{i,j}q^t_{i,j} \leq \left(\frac{2\epsilon/b}{1-\mathrm{e}^{-2\epsilon/b}}\right) bn\ln(n) + \left(\frac{2\epsilon/b}{1-\mathrm{e}^{-2\epsilon/b}} - 1\right)\epsilon nT \ .$$

Proof: By Theorem 21 and Lemma 14, we have that for any $\eta \geq 2\epsilon/b$,

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} P^t_{i,j}\left(\frac{\epsilon - q^t_{i,j}}{\eta b}\right) \leq \left(\frac{1}{1-e^{-\eta}}\right)\left(n\ln n + \eta \min_{\sigma\in S_n}\sum_{t=1}^{T}\sum_{i=1}^{n}\left(\frac{\epsilon - q^t_{i,\sigma(i)}}{\eta b}\right)\right) \ .$$

Using the fact that P^t is doubly stochastic, this gives us

$$\frac{\epsilon nT}{\eta b} - \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{P^t_{i,j}q^t_{i,j}}{\eta b} \leq \left(\frac{1}{1-e^{-\eta}}\right)\left(n\ln n + \frac{\epsilon nT}{b} - \max_{\sigma\in S_n}\sum_{t=1}^{T}\sum_{i=1}^{n}\frac{q^t_{i,\sigma(i)}}{b}\right) \ ,$$

and multiplying through by ηb yields

$$\epsilon nT - \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} P^t_{i,j}q^t_{i,j} \leq \left(\frac{\eta}{1-\mathrm{e}^{-\eta}}\right)\left(bn\ln n + \epsilon nT - \max_{\sigma\in S_n}\sum_{t=1}^{T}\sum_{i=1}^{n} q^t_{i,\sigma(i)}\right) \ .$$

Finally, rearranging terms gives us

$$\begin{aligned}\left(\frac{\eta}{1-\mathrm{e}^{-\eta}}\right)\max_{\sigma\in S_n}\sum_{t=1}^{T}\sum_{i=1}^{n}q_{i,\sigma(i)}^{t}-\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n}P_{i,j}^{t}q_{i,j}^{t}\\ \leq\quad\left(\frac{\eta}{1-\mathrm{e}^{-\eta}}\right)bn\ln(n)+\left(\frac{\eta}{1-\mathrm{e}^{-\eta}}-1\right)\epsilon nT\ .\end{aligned}$$

Notice that $\eta/(1-\mathrm{e}^{-\eta})\geq 1$ for positive values of η. Furthermore, $\eta/(1-\mathrm{e}^{-\eta})$ is strictly increasing in η. Thus the right hand side of this equation decreases as η decreases to 0. Setting $\eta=2\epsilon/b$ (the minimum value that η can take on while guaranteeing that the instantaneous loss is always in $[0,1]$) yields the result. ■

Let us examine the bound in the theorem. Notice that the term $(2\epsilon/b)/(1-\mathrm{e}^{-2\epsilon/b})$ goes to 1 in the limit as ϵ approaches 0. Additionally, the number of steps T scales inversely with ϵ since each lump purchase of q shares must be broken into $\lceil q/\epsilon\rceil$ individual purchases. Thus in the limit as ϵ approaches 0, the loss of the market maker is bounded by $bn\ln(n)$.

Corollary 5 *For any sequence of valid subset betting purchases broken into T $(=O(1/\epsilon))$ small purchases such that $q_{i,j}^{t}\in[-\epsilon,\epsilon]$ for all t, i, and j, let $P^1,\cdots,P^T$ be the price matrices obtained by running the Subset Betting Approximation Algorithm. In the limit as ϵ approaches 0,*

$$\max_{\sigma\in S_n}\sum_{t=1}^{T}\sum_{i=1}^{n}q_{i,\sigma(i)}^{t}-\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n}P_{i,j}^{t}q_{i,j}^{t}\leq bn\ln(n)\ .$$

This bound is comparable to worst-case loss bounds achieved using alternate methods for operating LMSRs on permutations. A single LMSR operated on the entire outcome space has a guaranteed worst-case loss of $b\ln(n!)$, but is, of course, intractable to operate. A set of n LMSRs operated as n separate markets, one for each position, would also have a total worst-case loss $bn\ln(n)$, but could not guarantee consistent prices. In the limit, our approximation algorithm achieves the same worst-case loss guarantee as if we were operating n separate markets, but prices remain consistent at all times.

5.7 Open Questions

In the previous section, we demonstrated an interesting and previously unexplored relationship between LMSR market makers and a common class of expert learning algorithms. However, it is likely that the connection between no-regret learning and prediction markets is deeper than we have suggested. Both can be viewed as techniques for aggregating the collective knowledge of many individuals by cleverly maintaining weights. Discovering additional connections could provide economists with the opportunity to take advantage of the already vast literature on online learning, and would likely benefit machine learning research as well. This is discussed more in Chapter 6.

Chapter 6

Future Directions

Significant further research is required before our foundational understanding of problems in collective learning will be sufficient to explain and resolve all of the issues faced by practitioners today. In this chapter, we conclude with a brief overview of some of the exciting areas of research that remain open.

6.1 Improved Models for Collaborative Filtering

The research described in Chapter 2 can be viewed as a theoretical foundation for rudimentary collaborative filtering. The collaborative filtering systems used in practice (e.g., the current Netflix movie recommendation system) are significantly more complex, but are often based on conglomerations of ad hoc techniques [13, 14]. While there have been some recent theoretical advancements on collaborative filtering, there is still a wide gap between our foundational understanding of the problem and algorithms that could perform well in practice.

Perhaps the most difficult aspect of closing this gap is developing the right model. To date, most work on collaborative filtering focuses on techniques for low-rank matrix completion [6, 117, 118], in which each preference rating is entered into a matrix, with rows representing users and columns representing objects (for example, websites). Missing entries in the matrix are approximated in such a way that the resulting matrix is of low rank. The assumption behind this line of work is that preferences can be decomposed into some small number of unknown factors. A significant drawback of this work is that it does not take into account is the existence of *known* features of

users (such as age or location) and objects (such as topic or designer, in the case of websites) that are often available in practice and could potentially be extremely useful. (One notable exception is the recent work of Abernethy et al. [2], which aims to incorporate known attributes of both users and products using specially designed attribute kernels, but does not provide theoretical guarantees about the quality of the solution.) When the amount of available training data is limited, it can be crucial to include these features; predicting a user's rating for a movie that no other user has rated is impossible without incorporating some outside information. On the other hand, the simple models we propose for taking known object features into account are not yet powerful enough to work well in practice.

In order to make real progress in this area, it seems necessary to define models that are able to draw on the power of matrix completion techniques while taking into account all available outside information. How to best define these models and what is provable in these settings remain open questions.

6.2 Network Diffusion and Viral Marketing

Sociologists have long been interested in the question of how new trends, behaviors, and innovations spread through social networks. This topic, known as *network diffusion*, has recently gained momentum in computer science due to the availability of data from social networks induced by online recommendation systems, social networking websites, and instant messaging systems [132]. These networks are significantly larger than any that have been studied before, in some cases containing hundreds of millions of nodes and more than a billion edges [93], so efficiency is a real concern.

One specific question that has received a lot of attention is how to determine the optimal group of individuals in a social network to target with an advertising campaign in order to cause a new product or technology to spread throughout the network. Kempe et al. [84, 85] proved that solving one natural variant of this problem exactly is NP-hard, but provided a simple greedy algorithm that can be used to obtain approximate results. However, their algorithm requires that all relevant parameters of the model are known, including the parameters describing the behavior of every individual in the network. They do not discuss techniques for learning these parameters from data

or the potential harm that might be caused by running their algorithm using incorrect parameter values.

Some effort has been made to use techniques from machine learning to learn the parameters necessary to approximately solve alternate variations of the question above [48, 108]. However, this work is based largely on heuristic techniques, and the authors make no attempt to show either that these heuristics produce accurate parameter estimates or that using inaccurate estimates doesn't have an adverse effect.

In working towards algorithms that are applicable in real viral marketing settings, it seems that an important next step is developing principled algorithms for learning or estimating the relevant modeling parameters without requiring too much expensive data or making unrealistic assumptions. The resulting algorithms would be of immediate interest to advertisers, and could have direct applications to many of the other problems to which ideas from the diffusion literature are commonly applied, potentially including preventing the spread of disease or contamination [95].

6.3 Social Search and Advertising

Most of the revenue of companies such as Google, Yahoo!, and Facebook comes from their ability to target ads to specific groups of users. Both search results and ads can be personalized based on user demographics such as age, gender, and location, as well as inferred user interests and online histories [120, 51, 71]. It is natural to speculate that when information about friendships and other network connections is available, it should be possible to use this information to build more complete user profiles by taking advantage of the assumption that people are likely to be similar to their friends (or at least likely to be interested in the same websites or products). However, to the best of our knowledge, there are no existing models of search or search advertising that capture the effects of the availability of network contacts and what can be learned from this extra social data.

There are a number of difficulties in building such a model. First, it is hard to know how much influence friends have over each other. A typical Facebook user might have hundreds of network "friends", but only a few with similar taste in books or movies, or whose opinions on products they fully trust. Even if it can be determined that two friends tend to listen to similar music or read the same books, there is an issue of causation to sort out; do people choose books based

on recommendations from friends, or do they choose friends who have similar interests and are therefore more likely to read the same books anyway? Finally, there are privacy issues that must be addressed. Privacy is always a concern with personalized search and advertising [89], but becomes even more important and complex when the personal information collected from a user can impact not only his own search results or ads but also those seen by his friends.

Despite these potential road blocks, the possibility of social search and advertising is worth some thought. In addition to posing a number of interesting theoretical questions, any ideas developed would have a clear impact on industry and, if done well, could improve the experience of web users too.

6.4 Additional Connections Between Learning and Markets

Section 5.6 details the first formal study of the connection between learning from expert advice and pricing algorithms for prediction markets. Both areas can be viewed as studies of how to aggregate the knowledge of many potentially diverse individuals. Furthermore, in both cases, this aggregation is accomplished by cleverly maintaining sets of weights, either over outcomes (in the case of prediction markets) or over members of the population themselves (in the case of learning from expert advice). Despite these apparent similarities, no formal connections between the two areas have been posed beyond those described in this work.

The discovery and understanding of additional connections between machine learning and prediction markets could have huge impact. This somewhat open-ended line of work has the potential to give researchers studying both machine learning and market design the opportunity to learn from and take advantage of years of existing research and prior knowledge, potentially leading to breakthroughs in both fields, and should definitely be explored further.

Appendix

A1 Basic Tools from Probability Theory

For the sake of completeness, in this section we provide two simple results from probability theory that are used throughout this document.

A1.1 Hoeffding's Inequality

Hoeffding's inequality [69] is a standard result in probability theory that can be used to bound the probability that the sum of a set of random variables is far from its expectation. Here we state a general form of the inequality, which is used frequently throughout this dissertation.

Theorem 23 (Hoeffding's Inequality [69]) *Let $x_1, \cdots, x_n$ be n independent random variables, with $x_i \in [a_i, b_i]$ for each i. Then for any value t,*

$$\Pr\left[\left|\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} E[x_i]\right| \geq tn\right] \leq 2\exp\left(\frac{-2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) .$$

Notice that it is *not* required that $x_1, \cdots, x_n$ are identically distributed. If each $x_i \in [0, 1]$, then the expression simplifies to

$$\Pr\left[\left|\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} E[x_i]\right| \geq tn\right] \leq 2\mathrm{e}^{-2nt^2} .$$

A1.2 McDiarmid's Inequality

McDiarmid's inequality [102] is a generalization of Hoeffding's inequality that can be used to bound the probability that a function depending on many independent random variables is far from

its expectation as long as the function doesn't depend too much on any single variable.

Theorem 24 (McDiarmid's Inequality [102]) *Let $x_1, \cdots, x_n$ be independent random variables taking on values in a set A and assume that $f : A^n \to \mathbb{R}$ satisfies*

$$\sup_{x_1,\ldots,x_n,x_i' \in A} |f(x_1,\ldots,x_n) - f(x_1,\ldots,x_{i-1},x_{i'},x_{i+1},\ldots,x_n)| \leq c_i$$

for every $1 \leq i \leq n$. Then for every $t > 0$,

$$\Pr\left[|f(x_1,\ldots,x_n) - \mathrm{E}\left[f(x_1,\ldots,x_n)\right]| \geq t\right] \leq 2\exp\left(\frac{-2t^2}{\sum_{i=1}^n c_i^2}\right) .$$

A2 Additional Proofs from Chapter 2

A2.1 Proof of Lemma 3

Here we show one direction of the bound, namely that with probability $1 - \delta/2$, for all $h \in \mathcal{H}$,

$$e(h) \leq \hat{e}(h) + 2LR_n(\mathcal{H}) + \sqrt{\frac{2\ln(2/\delta)}{n}} .$$

The proof of the other direction is nearly identical. For $i \in \{1,\ldots,n\}$, let $\langle x_i, y_i\rangle$ be the ith training instance, distributed according to P_i, and let $\langle x_i', y_i'\rangle$ be independent random variables drawn according to P_i. Note that for all $h \in \mathcal{H}$,

$$\begin{aligned}
e(h) &= e(h) + \hat{e}(h) - \hat{e}(h) \leq \hat{e}(h) + \sup_{h' \in \mathcal{H}} \left(e(h') - \hat{e}(h')\right) \\
&= \hat{e}(h) + \sup_{h' \in \mathcal{H}} \left(\mathrm{E}_{\{\langle x_i', y_i'\rangle\}_{i=1}^n} \left[\frac{1}{n}\sum_{i=1}^n \phi(y_i', h'(x_i'))\right] - \frac{1}{n}\sum_{i=1}^n \phi(y_i, h'(x_i)) \right) \\
&= \hat{e}(h) + \sup_{h' \in \mathcal{H}} \left(\mathrm{E}_{\{\langle x_i', y_i'\rangle\}_{i=1}^n} \left[\frac{1}{n}\sum_{i=1}^n \phi'(y_i', h'(x_i')) + \phi(y_i', 0)\right] \right. \\
&\qquad \left. -\frac{1}{n}\sum_{i=1}^n \phi'(y_i, h'(x_i)) + \phi(y_i, 0) \right) .
\end{aligned}$$

When only one instance $\langle x_i, y_i\rangle$ changes, the $\sup$ term can change by at most $2/n$. Thus we

can apply McDiarmid's inequality (see Section A1.2) to see that with probability at least $1 - \delta/2$,

$$e(h) \leq \hat{e}(h) + \mathrm{E}\left[\sup_{h' \in \mathcal{H}} \left(\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n} \phi'(y_i', h'(x_i'))\right] - \frac{1}{n}\sum_{i=1}^{n} \phi'(y_i, h'(x_i))\right)\right] + \sqrt{\frac{2\ln(2/\delta)}{n}} \,,$$

where the outer expectation is with respect to set of training instances $\{\langle x_i, y_i\rangle\}_{i=1}^n$ and the inner expectation is with respect to the set of random variables $\{\langle x_i', y_i'\rangle\}_{i=1}^n$. Now it suffices to show that this middle term is bounded by $2LR_n(\mathcal{H})$. Using the fact that the supremum of an expectation is less than or equal to the expectation of a supremum, we find that

$$\begin{aligned}
&\mathrm{E}_{\{\langle x_i, y_i\rangle\}_{i=1}^n}\left[\sup_{h' \in \mathcal{H}} \left(\mathrm{E}_{\{\langle x_i', y_i'\rangle\}_{i=1}^n}\left[\frac{1}{n}\sum_{i=1}^{n} \phi'(y_i', h'(x_i'))\right] - \frac{1}{n}\sum_{i=1}^{n} \phi'(y_i, h'(x_i))\right)\right] \\
&\quad \leq \mathrm{E}_{\{\langle x_i, y_i\rangle\}_{i=1}^n, \{\langle x_i', y_i'\rangle\}_{i=1}^n}\left[\sup_{h' \in \mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \left(\phi'(y_i', h'(x_i')) - \phi'(y_i, h'(x_i))\right)\right] \\
&\quad = \mathrm{E}_{\{\langle x_i, y_i\rangle\}_{i=1}^n, \{\langle x_i', y_i'\rangle\}_{i=1}^n, \{\sigma_i\}_{i=1}^n}\left[\sup_{h' \in \mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i\left(\phi'(y_i', h'(x_i')) - \phi'(y_i, h'(x_i))\right)\right] \\
&\quad \leq \mathrm{E}_{\{\langle x_i, y_i\rangle\}_{i=1}^n, \{\sigma_i\}_{i=1}^n}\left[\sup_{h' \in \mathcal{H}} \frac{2}{n}\sum_{i=1}^{n} \sigma_i \phi'(y_i, h'(x_i))\right] = R_n(\phi' \circ \mathcal{H}) \,.
\end{aligned}$$

Lemma 2 implies that $R_n(\phi' \circ \mathcal{H}) \leq 2LR_n(\mathcal{H})$ since ϕ is Lipschitz with parameter L. The result follows.

A2.2 Proof of Lemma 5

We cannot apply Lemma 3 directly using the squared loss function, since it may output values outside of the range $[0, 1]$. Instead, we apply the Lemma 3 using the alternate loss function $\mathcal{L}'(h, \langle x, y\rangle) = \phi(y, h(x))$ where

$$\phi(y, a) = \begin{cases} \frac{1}{4B^2}(y + B)^2 & \text{if } a < -B, \\ \frac{1}{4B^2}(y - a)^2 & \text{if } -B \leq a \leq B, \\ \frac{1}{4B^2}(y + B)^2 & \text{if } a > B. \end{cases}$$

It is easy to see that ϕ always outputs values in the range $[0, 1]$. Furthermore, for any $y \in [-B, B]$, ϕ is Lipschitz in the second parameter with parameter $1/B$. For any $[a, b] \in [-B, B]$,

$$\begin{aligned} |\phi(y,a) - \phi(y,b)| &= \frac{1}{4B^2}\left|(y-a)^2 - (y-b)^2\right| = \frac{1}{4B^2}\left|a^2 - b^2 + 2y(b-a)\right| \\ &\leq \frac{1}{4B^2}\left|a^2 - b^2\right| + \frac{1}{2B^2}\left|y(a-b)\right| \\ &\leq \frac{1}{4B^2}\left|a+b\right|\left|a-b\right| + \frac{1}{2B^2}\left|y(a-b)\right| \leq \frac{1}{B}\left|a-b\right| \ . \end{aligned}$$

Applying Lemma 3 gives a uniform convergence bound of $(2/B)R_n(\mathcal{H}) + \sqrt{2\ln(2/\delta)/n}$ for $\mathcal{L}'$. Scaling by $4B^2$ yields the bound for $\mathcal{L}$.

A2.3 Proof of Theorem 5

The proof first requires a uniform convergence bound for the empirical $\boldsymbol{\alpha}$-error, which is given in the following lemma. Note that this bound is minimized when α_i is proportional to n_i. In other words, convergence is fastest when all data instances are weighted equally.

Lemma 15 *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d. If a random labeled sample is generated by drawing n_i points from each distribution from $\mathcal{D}_i$, and labeling them according to f_i, then with probability at least $1-\delta$, for every $h \in \mathcal{H}$:*

$$|\hat{e}_{\boldsymbol{\alpha}}(h) - e_{\boldsymbol{\alpha}}(h)| \leq \sqrt{\sum_{i=1}^{K} \frac{\alpha_i^2}{2n_i}\left(d\log(2n_{1:K}) + \log(1/\delta)\right)} \ .$$

Proof: For each source i, let $X_{i,1}, \ldots, X_{i,n_i}$ be random variables that take on the values

$$\frac{\alpha_i n_{1:K}}{n_i}|h(x) - f_i(x)|$$

for the n_i instances $x \in S_i$. Note that $X_{i,1}, \ldots, X_{i,n_i} \in [0, \alpha_i n_{1:K}/n_i]$. Then

$$\hat{e}_{\boldsymbol{\alpha}}(h) = \sum_{i=1}^{K} \alpha_i \hat{e}_i(h) = \sum_{i=1}^{K} \alpha_i \frac{1}{n_i} \sum_{x \in S_i} |h(x) - f_i(x)| = \frac{1}{n_{1:K}} \sum_{i=1}^{K} \sum_{j=1}^{n_i} X_{i,j}.$$

By linearity of expectations, we have that $E[\hat{e}_{\boldsymbol{\alpha}}(h)] = e_{\boldsymbol{\alpha}}(h)$, and so by Hoeffding's inequality, for every $h \in \mathcal{H}$,

$$\Pr\left[|\hat{e}_{\boldsymbol{\alpha}}(h) - e_{\boldsymbol{\alpha}}(h)| \geq \epsilon\right] \leq 2\exp\left(\frac{-2n_{1:K}^2\epsilon^2}{\sum_{i=1}^{K}\sum_{j=1}^{n_i}\operatorname{range}^2(X_{i,j})}\right) = 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{K}\frac{\alpha_i^2}{n_i}}\right).$$

The remainder of the proof for hypothesis classes of finite VC dimension follows a standard argument. In particular, the reduction to a finite hypothesis class using the growth function does not change [125, 4]. This, combined with the union bound, gives us the probability that there exists *any* hypothesis $h \in \mathcal{H}$ such that $|\hat{e}_{\alpha}(h) - e_{\alpha}(h)| \geq \epsilon$. Substituting δ for the probability and solving for ϵ gives us the bound. ■

We are now ready to state the proof of Theorem 5.

Let $h_i^* = \operatorname{argmin}_h\{e_T(h) + e_i(h)\}$. For each source i (and similarly for the target T), define $e_i(h, h') = \mathrm{E}_{x\sim\mathcal{D}_i}\left[|h(x) - h'(x)|\right]$. Then

$$\begin{aligned}
|e_{\boldsymbol{\alpha}}(h) - e_T(h)| &= \left|\sum_{i=1}^{K}\alpha_i e_i(h) - e_T(h)\right| \leq \sum_{i=1}^{K}\alpha_i\left|e_i(h) - e_T(h)\right| \\
&\leq \sum_{i=1}^{K}\alpha_i\left(|e_i(h) - e_i(h, h_i^*)| + |e_i(h, h_i^*) - e_T(h, h_i^*)|\right. \\
&\qquad \left. + |e_T(h, h_i^*) - e_T(h)|\right) \\
&\leq \sum_{i=1}^{K}\alpha_i\left(e_i(h_i^*) + |e_i(h, h_i^*) - e_T(h, h_i^*)| + e_T(h_i^*)\right) \\
&\leq \sum_{i=1}^{K}\alpha_i\left(\min_{h\in\mathcal{H}}\{e_T(h) + e_i(h)\} + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_T)\right).
\end{aligned}$$

The third line follows from the triangle inequality. The last line follows from the simple fact that for any hypotheses $h, h' \in \mathcal{H}$, for any source S_i,

$$|e_S(h, h') - e_T(h, h')| \leq \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T).$$

For notational convenience, let $\lambda_i = \min_{h\in\mathcal{H}}\{e_T(h)+e_i(h)\}$. Putting this together with Lemma 15, we find that for any $\delta\in(0,1)$, with probability $1-\delta$,

$$\begin{aligned}
e_T(\hat{h}) &\leq e_{\boldsymbol{\alpha}}(\hat{h}) + \sum_{i=1}^{K}\alpha_i\left(\lambda_i + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i,D_T)\right) \\
&\leq \hat{e}_{\boldsymbol{\alpha}}(\hat{h}) + \sqrt{\sum_{i=1}^{K}\frac{\alpha_i^2}{2n_i}\big(d\log(2n_{1:K})+\log(1/\delta)\big)} + \sum_{i=1}^{K}\alpha_i\left(\lambda_i + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i,D_T)\right) \\
&\leq \hat{e}_{\boldsymbol{\alpha}}(h_T^*) + \sqrt{\sum_{i=1}^{K}\frac{\alpha_i^2}{2n_i}\big(d\log(2n_{1:K})+\log(1/\delta)\big)} + \sum_{i=1}^{K}\alpha_i\left(\lambda_i + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i,D_T)\right) \\
&\leq e_{\boldsymbol{\alpha}}(h_T^*) + 2\sqrt{\sum_{i=1}^{K}\frac{\alpha_i^2}{2n_i}\big(d\log(2n_{1:K})+\log(1/\delta)\big)} + \sum_{i=1}^{K}\alpha_i\left(\lambda_i + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i,D_T)\right) \\
&\leq e_T(h_T^*) + 2\sqrt{\sum_{i=1}^{K}\frac{\alpha_i^2}{2n_i}\big(d\log(2n_{1:K})+\log(1/\delta)\big)} + \sum_{i=1}^{K}\alpha_i(2\lambda_i + d_{\mathcal{H}\Delta\mathcal{H}}(D_i,D_T)) \ .
\end{aligned}$$

A2.4 Proof of Theorem 6

The proof is almost identical to that of Theorem 5 with minor modifications to the derivation of the bound on $|e_{\boldsymbol{\alpha}}(h)-e_T(h)|$. Let $h^* = \operatorname{argmin}_h\{e_T(h)+e_{\boldsymbol{\alpha}}(h)\}$. By the triangle inequality,

$$\begin{aligned}
|e_{\boldsymbol{\alpha}}(h)-e_T(h)| &\leq |e_{\boldsymbol{\alpha}}(h)-e_{\boldsymbol{\alpha}}(h,h^*)| + |e_{\boldsymbol{\alpha}}(h,h^*)-e_T(h,h^*)| + |e_T(h,h^*)-e_T(h)| \\
&\leq e_{\boldsymbol{\alpha}}(h^*) + |e_{\boldsymbol{\alpha}}(h,h^*)-e_T(h,h^*)| + e_T(h^*) \\
&\leq \min_h\left\{e_T(h)+\sum_{i=1}^{K}\alpha_i e_i(h)\right\} + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_{\boldsymbol{\alpha}},D_T) \ .
\end{aligned}$$

The remainder of the proof is unchanged.

A3 Additional Proofs from Chapter 3

A3.1 Proof Sketch of Theorem 7

We first sketch the hardness construction. Let $\mathcal{H}$ be any class of Boolean circuits (that is, with gates in $\mathcal{C}$) that is not polynomially learnable in the standard PAC model; under standard cryptographic assumptions, such a class exists. Let D be a hard distribution for PAC learning $\mathcal{H}$. Let $h\in\mathcal{H}$

be a Boolean circuit with R inputs, S gates, and depth D. To embed the computation by h in a collective problem, we let $N = R + S$ and $T = D$. We introduce an agent for each of the R inputs to h, whose value after the initial state is set according to an arbitrary AND, OR, or NOT gate. We additionally introduce one agent for every gate g in h. If a gate g in h takes as its inputs the outputs of gates g' and g'', then at each time step the agent corresponding to g computes the corresponding function of the states of the agents corresponding to g' and g'' at the previous time step. Finally, by convention we always have the Nth agent be the agent corresponding to the output gate of h, and define the output function as $F(\mathbf{s}) = s_N$. The distribution P over initial states of the N agents is identical to D on the R agents corresponding to the inputs of h, and arbitrary (e.g., independent and uniform) on the remaining S agents.

Despite the fact that this construction introduces a great deal of spurious computation (for instance, at the first time step, many or most gates may simply be computing Boolean functions of the random bits assigned to non-input agents), it is clear that if gate g is at depth d in h, then at time d in the collective simulation of the agents, the corresponding agent has exactly the value computed by g under the inputs to h (which are distributed according to D). Because the outcome function is the value of the agent corresponding to the output gate of h at time $T = D$, pairs of the form $\left\langle \mathbf{s}^0, F(\mathbf{s}^T) \right\rangle$ provide exactly the same data as the PAC model for h under D, and thus must be equally hard.

For the polynomial learnability of $\mathcal{C}$ from collective behavior, we note that $\mathcal{C}$ is clearly PAC learnable, since it is just Boolean combinations of 1 or 2 inputs. In Section 3.4 we give a general reduction from collective learning of any agent strategy class to PAC learning the class, thus giving the claimed result.

A3.2 Proof of Lemma 8

We bound the error of these estimations in two parts. First, since from Equation 3.6 we know that for any $\delta_1 > 0$, with probability $1 - \delta_1$,

$$\left| \hat{\alpha} \sum_{m:\mathcal{I}_m \in \mathcal{M}} f_{m,a} - \alpha \sum_{m:\mathcal{I}_m \in \mathcal{M}} f_{m,a} \right| \leq \frac{1}{Z_{a^*}} \sqrt{\frac{\ln(4/\delta_1)}{M}} \sum_{m:\mathcal{I}_m \in \mathcal{M}} f_{m,a} \leq \frac{1}{Z_{a^*}} \sqrt{M \ln(4/\delta_1)} \, ,$$

and

$$|(1-\hat{\alpha})M - (1-\alpha)M| \leq \frac{1}{Z_{a^*}}\sqrt{M\ln(4/\delta_1)}\,,$$

we have by Lemma 9 that for sufficiently large M

$$\begin{aligned}
&\left|\frac{\hat{\alpha}\sum_{m:\mathcal{I}_m\in\mathcal{M}} f_{m,a}}{(1-\hat{\alpha})M} - \frac{\alpha\sum_{m:\mathcal{I}_m\in\mathcal{M}} f_{m,a}}{(1-\alpha)M}\right| \\
&\quad\leq \frac{(1/Z_{a^*})\sqrt{M\ln(4/\delta_1)}\left((1-\hat{\alpha})M + \hat{\alpha}\sum_{m:\mathcal{I}_m\in\mathcal{M}} f_{m,a}\right)}{(1-\hat{\alpha})^2M^2 - (1-\hat{\alpha})M\sqrt{M\ln(4/\delta_1)}/Z_{a^*}} \\
&\quad\leq \frac{\sqrt{M\ln(4/\delta_1)}((1-\hat{\alpha})M + \hat{\alpha}M)}{Z_{a^*}(1-\hat{\alpha})^2M^2 - (1-\hat{\alpha})M\sqrt{M\ln(4/\delta_1)}} \\
&\quad= \frac{\sqrt{\ln(4/\delta_1)}}{Z_{a^*}(1-\hat{\alpha})^2\sqrt{M} - (1-\hat{\alpha})\sqrt{\ln(4/\delta_1)}}\,.
\end{aligned}$$

Now, by Hoeffding's inequality and the union bound, for any $\delta_2 > 0$, with probability $1-\delta_2$, for all a,

$$\left|\sum_{m:\mathcal{I}_m\in\mathcal{M}} \mathrm{I}(a_m = a) - \mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}} \mathrm{I}(a_m = a)\right]\right| \leq \sqrt{M\ln(2K/\delta_2)/2}\,.$$

Setting $\delta_2 = K\delta_1/2$, we can again apply Lemma 9 and see that for sufficiently large M

$$\begin{aligned}
&\left|\frac{\sum_{m:\mathcal{I}_m\in\mathcal{M}} \mathrm{I}(a_m = a)}{(1-\hat{\alpha})M} - \frac{\mathrm{E}\left[\sum_{m:\mathcal{I}_m\in\mathcal{M}} \mathrm{I}(a_m = a)\right]}{(1-\alpha)M}\right| \\
&\quad\leq \frac{\sqrt{M\ln(4/\delta_1)/2}\left((1-\hat{\alpha})Z_{a^*}M + \sqrt{2}M\right)}{Z_{a^*}(1-\hat{\alpha})^2M^2 - (1-\hat{\alpha})M\sqrt{M\ln(4/\delta_1)}} \\
&\quad\leq \frac{\sqrt{\ln(4/\delta_1)/2}\left((1-\hat{\alpha})Z_{a^*} + \sqrt{2}\right)}{Z_{a^*}(1-\hat{\alpha})^2\sqrt{M} - (1-\hat{\alpha})\sqrt{\ln(4/\delta_1)}} = \frac{((1-\hat{\alpha})Z_{a^*}/\sqrt{2}+1)\sqrt{\ln(4/\delta_1)}}{Z_{a^*}(1-\hat{\alpha})^2\sqrt{M} - (1-\hat{\alpha})\sqrt{\ln(4/\delta_1)}}\,.
\end{aligned}$$

Setting $\delta_1 = \delta/(1+K/2)$ and applying the union bound yields the lemma.

A3.3 Proof of Lemma 9

For the first direction,

$$\begin{aligned}\frac{u}{v}-\frac{\hat{u}}{\hat{v}} &\leq \frac{u}{v}-\frac{u-\epsilon}{v+k\epsilon}=\frac{u}{v}-\frac{uv-\epsilon v}{v(v+k\epsilon)}=\frac{u}{v}-\frac{uv+uk\epsilon-\epsilon v-uk\epsilon}{v(v+k\epsilon)}\\ &= \frac{u}{v}-\frac{u(v+k\epsilon)-\epsilon(v+uk)}{v(v+k\epsilon)}=\frac{\epsilon(v+uk)}{v(v+\epsilon k)}\leq\frac{\epsilon(v+uk)}{v(v-\epsilon k)}\,.\end{aligned}$$

Similarly,

$$\begin{aligned}\frac{\hat{u}}{\hat{v}}-\frac{u}{v} &\leq \frac{u+\epsilon}{v-k\epsilon}-\frac{u}{v}=\frac{uv+\epsilon v}{v(v-k\epsilon)}-\frac{u}{v}=\frac{uv-uk\epsilon+\epsilon v+uk\epsilon}{v(v-k\epsilon)}-\frac{u}{v}\\ &= \frac{u(v-k\epsilon)+\epsilon(v+uk)}{v(v-k\epsilon)}-\frac{u}{v}=\frac{\epsilon(v+uk)}{v(v-\epsilon k)}\,.\end{aligned}$$

A3.4 Proof of Lemma 10

As long as M is sufficiently large, for any fixed $\mathbf{f}$,

$$\begin{aligned}&\sum_{a\in\mathcal{S}}|(\alpha f_a+(1-\alpha)w_a)-(\hat{\alpha}f_a+(1-\hat{\alpha})\hat{w}_a)|\\ &\leq \sum_{a\in\mathcal{S}}|\alpha-\hat{\alpha}|f_a+\sum_{a\in\mathcal{S}}|(1-\alpha)w_a-(1-\hat{\alpha})\hat{w}_a|\\ &\leq |\alpha-\hat{\alpha}|+\sum_{a\in\mathcal{S}}\left(|\alpha-\hat{\alpha}|\hat{w}_a+|w_a-\hat{w}_a|(1-\hat{\alpha})-|\alpha-\hat{\alpha}|\cdot|w_a-\hat{w}_a|\right)\\ &\leq 2|\alpha-\hat{\alpha}|+(1-\hat{\alpha})\sum_{a\in\mathcal{S}}|w_a-\hat{w}_a|-|\alpha-\hat{\alpha}|\sum_{a\in\mathcal{S}}|w_a-\hat{w}_a|\\ &\leq \frac{2\sqrt{\ln((4+2K)/\delta)}}{Z_{a^*}\sqrt{M}}+\min\left\{\frac{K(Z_{a^*}/\sqrt{2}+2)\sqrt{\ln((4+2K)/\delta)}}{Z_{a^*}(1-\hat{\alpha})\sqrt{M}-\sqrt{\ln((4+2K)/\delta)}},2(1-\hat{\alpha})\right\}\,.\end{aligned}$$

Notice that this holds uniformly for all $\mathbf{f}$, so the same bound holds when we take an expectation over $\mathbf{f}$.

A3.5 Handling the case where Z_{a^*} is small

Suppose that for an action a, $Z_a<\epsilon$. Let η_a and μ_a be the true median and mean respectively of the distribution from which the random variables $f_{m,a}$ are drawn. Let f_a^{high} be the mean value of the distribution over $f_{m,a}$ conditioned on $f_{m,a}>\eta_a$, and similarly let f_a^{low} be the mean value

conditioned on $f_{m,a} < \eta_a$, so $\mu_a = \left(f_a^{low} + f_a^{high}\right)/2$.[1] Let $\bar{f}_a^{high}$ be the empirical average of $f_{m,a}$ conditioned on $f_{m,a} > \eta_a$, and $\bar{f}_a^{low}$ be the empirical average of $f_{m,a}$ conditioned on $f_{m,a} < \eta_a$. (Note that we cannot actually compute $\bar{f}_a^{high}$ and $\bar{f}_a^{low}$ since η_a is unknown.) Finally, let $\hat{f}_a^{high} = (2/M) \sum_{m:\mathcal{I}_m \in \mathcal{M}_a^{high}} f_{m,a}$ and $\hat{f}_a^{low} = (2/M) \sum_{m:\mathcal{I}_m \in \mathcal{M}_a^{low}} f_{m,a}$. Notice that $Z_a = \hat{f}_a^{high} - \hat{f}_a^{low}$.

We show first that f_a^{high} and f_a^{low} are close to $\bar{f}_a^{high}$ and $\bar{f}_a^{low}$ respectively. Next we show that $\bar{f}_a^{high}$ and $\bar{f}_a^{low}$ are close to $\hat{f}_a^{high}$ and $\hat{f}_a^{low}$ respectively. Finally, we show that this implies that if Z_a is small, then the probability that a random value of $f_{m,a}$ is far from η_a is small. This in turn implies a small $\mathcal{L}_1$ distance between our estimated model and the real model for each agent.

To bound the difference between f_a^{high} and $\bar{f}_a^{high}$, it is first necessary to lower bound the number of points in the empirical samples with $f_{m,a} > \eta_a$. Let z_m be a random variable that is 1 if $f_{m,a} > \eta_a$ and 0 otherwise. Clearly $\Pr[z_m = 1] = \Pr[z_m = 0] = 1/2$. By a straightforward application of Hoeffding's inequality, for any δ_3, with probability $1 - \delta_3$,

$$\left| \sum_{m:\mathcal{I}_m \in \mathcal{M}} z_m - \frac{M}{2} \right| \leq \sqrt{\frac{\ln(2/\delta_3)M}{2}} ,$$

and so the number of samples averaged to get $\bar{f}_a^{high}$ is at least $(M/2) - \sqrt{\ln(2/\delta_3)M/2}$. Applying Hoeffding's inequality again, with probability $1 - \delta_4$,

$$\left| f_a^{high} - \bar{f}_a^{high} \right| \leq \sqrt{\frac{\ln(2/\delta_4)}{M - \sqrt{2\ln(2/\delta_3)M}}} .$$

Now, $\hat{f}_a^{high}$ is an empirical average of $M/2$ values in $[0, 1]$, while $\bar{f}_a^{high}$ is an empirical average of the same points, plus or minus up to $\sqrt{\ln(2/\delta_3)M/2}$ points. In the worst case, $\bar{f}_a^{high}$ either includes an additional $\sqrt{\ln(2/\delta_3)M/2}$ points with values lower than any point in $\mathcal{M}_a^{high}$, or excludes the $\sqrt{\ln(2/\delta_3)M/2}$ lowest values points in $\mathcal{M}_a^{high}$. This implies that in the worst case,

$$\left| \bar{f}_a^{high} - \hat{f}_a^{high} \right| \leq \frac{\sqrt{\ln(2/\delta_3)}}{\sqrt{M/2} - \sqrt{\ln(2/\delta_3)}} .$$

[1] Assume for now that it is never the case that $f_{m,a} = \eta_a$. This simplifies the explanation, although everything still holds if $f_{m,a}$ can be η_a.

By the triangle inequality,

$$\left|f_a^{high} - \hat{f}_a^{high}\right| \leq \sqrt{\frac{\ln(2/\delta_4)}{M - \sqrt{2\ln(2/\delta_3)M}}} + \frac{\sqrt{\ln(2/\delta_3)}}{\sqrt{M/2} - \sqrt{\ln(2/\delta_3)}} \,.$$

The same can be show for f_a^{low} and $\hat{f}_a^{low}$. Hence if $Z_a \leq \epsilon$, then

$$f_a^{high} - f_a^{low} \leq \epsilon + 2\sqrt{\frac{\ln(2/\delta_4)}{M - \sqrt{2\ln(2/\delta_3)M}}} + \frac{2\sqrt{\ln(2/\delta_3)}}{\sqrt{M/2} - \sqrt{\ln(2/\delta_3)}} \,.$$

Call this quantity ϵ'. Clearly we have

$$\mu_a - \epsilon' \leq f_a^{low} \leq \mu_a \leq f_a^{high} \leq \mu_a + \epsilon'.$$

Since f_a^{high} is an average of points which are all higher than f_a^{low}, and similarly f_a^{low} is an average of points all lower than f_a^{high}, this implies that for any $\tau \geq 0$,

$$\Pr\left[|f_{m,a} - \mu_a| \geq \epsilon' + \tau\right] \leq \Pr\left[f_{m,a} \geq f_a^{high} + \tau\right] + \Pr\left[f_{m,a} \leq f_a^{low} - \tau\right] \leq \epsilon'/(\epsilon' + \tau) \,.$$

Recall that when Z_a is small for all a, we set $\hat{\alpha} = 0$ and $\hat{w}_a = \sum_{m:\mathcal{I}_m \in \mathcal{M}} \mathrm{I}(a_m = a)$. Let $\bar{w}_a = \alpha\mu_a + (1-\alpha)w_a$. Notice that $\mathrm{E}\left[\hat{w}_a\right] = \bar{w}_a$. Applying Hoeffding's inequality yet again, with probability $1 - \delta_5$, $|\hat{w}_a - \bar{w}_a| \leq \sqrt{\ln(2/\delta_5)/2M}$. For any $\tau > 0$,

$$\begin{aligned}
&\mathrm{E}_{\mathbf{f}\sim D^f}\left[\sum_{a\in\mathcal{S}} |(\alpha f_a + (1-\alpha)w_a) - (\hat{\alpha} f_a + (1-\hat{\alpha})\hat{w}_a)|\right] \\
&= \sum_{a\in\mathcal{S}} \mathrm{E}_{\mathbf{f}\sim D^f}\left[|\alpha f_a + (1-\alpha)w_a - \hat{w}_a|\right] \\
&\leq \sum_{a\in\mathcal{S}} \mathrm{E}_{\mathbf{f}\sim D^f}\left[|\alpha f_a + (1-\alpha)w_a - \bar{w}_a| + |\bar{w}_a - \hat{w}_a|\right] \\
&\leq \alpha \sum_{a\in\mathcal{S}} \mathrm{E}_{\mathbf{f}\sim D^f}\left[|f_a - \mu_a|\right] + K\sqrt{\ln(2/\delta_5)/2M} \\
&\leq K\left(\left(1 - \frac{\epsilon'}{\epsilon'+\tau}\right)(\epsilon'+\tau) + \frac{\epsilon'}{\epsilon'+\tau}\right) + K\sqrt{\ln(2/\delta_5)/2M} \\
&= K\left(\tau + \frac{\epsilon'}{\epsilon'+\tau} + \sqrt{\ln(2/\delta_5)/2M}\right) \,.
\end{aligned}$$

A3.6 Proof of Lemma 12

By the union bound,

$$\begin{aligned}&\Pr_{\mathbf{f}\sim D^f}\left[\exists a,b\in\mathcal{S}: f_a>0, f_b>0, |\mathcal{M}_{a,b}|<M\right]\\ &\quad\leq \sum_{a,b\in\mathcal{S}}\Pr_{\mathbf{f}\sim D^f}\left[f_a>0,f_b>0,|\mathcal{M}_{a,b}|<M\right]\leq \sum_{a,b\in\mathcal{S}:|\mathcal{M}_{a,b}|<M}\Pr_{\mathbf{f}\sim D^f}\left[f_a>0,f_b>0\right]\ .\end{aligned}$$

For any fixed pair of actions a, b such that $|\mathcal{M}_{a,b}| < M$, it follows from Hoeffding's inequality that for any $\delta \in (0,1)$, with probability $1-\delta$,

$$\Pr_{\mathbf{f}\sim D^f}\left[f_a>0,f_b>0\right]\leq\frac{M}{|\mathcal{M}|}+\sqrt{\frac{\ln(1/\delta)}{2|\mathcal{M}|}}\ .$$

Noting that the number of pairs is less than $K^2/2$ and setting $\delta = 2\delta/K^2$ yields the lemma.

A3.7 Bounding the $\mathcal{L}_1$

Here we show that with high probability (over the choice of $\mathcal{M}$ and the draw of $\mathbf{f}$), if $\mathcal{M}$ is sufficiently large,

$$\sum_{a\in\mathcal{S}}\left|\frac{w_a f_a}{\sum_{s\in\mathcal{S}}w_s f_s}-\frac{\hat{w}_a f_a}{\sum_{s\in\mathcal{S}}\hat{w}_s f_s}\right| \quad\leq\quad \frac{2(1+\beta)KN\sqrt{N\ln(2K/\delta)}}{\sqrt{2M}-(1+\beta)K(N+1)\sqrt{N\ln(2K/\delta)}}\ .$$

First note that we can rewrite the expression on the left-hand side of the inequality above as

$$\sum_{a\in\mathcal{S}'}\left|\frac{w'_a f_a}{\sum_{s\in\mathcal{S}'}w'_s f_s}-\frac{\hat{w}_a f_a}{\sum_{s\in\mathcal{S}'}\hat{w}_s f_s}\right|\ ,$$

where for all a, $w'_a = w_a/(\sum_{s\in\mathcal{S}'} w_s)$. We know from Equation 3.9 that with high probability (over the data set and the choice of $\mathbf{f}$), for all $a \in \mathcal{S}'$,

$$|w'_a f_a-\hat{w}_a f_a|\leq\frac{f_a(1+\beta)|\mathcal{S}'|\sqrt{N\ln(2K^2/\delta)}}{\sqrt{2M}-(1+\beta)|\mathcal{S}'|\sqrt{N\ln(2K^2/\delta)}}\ ,$$

and

$$\left|\sum_{s\in\mathcal{S}'} w'_s f_s - \sum_{s\in\mathcal{S}'} w_s f_s\right| \leq \sum_{s\in\mathcal{S}'} f_s|w'_s - \hat{w}_s| \leq \frac{(1+\beta)|\mathcal{S}'|\sqrt{N\ln(2K^2/\delta)}}{\sqrt{2M} - (1+\beta)|\mathcal{S}'|\sqrt{N\ln(2K^2/\delta)}}\,.$$

Thus, using the fact that $\sum_{s\in\mathcal{S}'} \hat{w}_s f_s \geq 1/N$, we can apply Lemma 9 once again to get that

$$\left|\frac{w'_a f_a}{\sum_{s\in\mathcal{S}'} w'_s f_s} - \frac{\hat{w}_a f_a}{\sum_{s\in\mathcal{S}'} \hat{w}_s f_s}\right| \leq \frac{(1+\beta)KN\sqrt{N\ln(2K/\delta)}\left(f_a + \frac{\hat{w}_a f_a}{\sum_{s\in\mathcal{S}'}\hat{w}_s f_s}\right)}{\sqrt{2M} - (1+\beta)K(N+1)\sqrt{N\ln(2K/\delta)}}\,,$$

and

$$\sum_{a\in\mathcal{S}'}\left|\frac{w'_a f_a}{\sum_{s\in\mathcal{S}'} w'_s f_s} - \frac{\hat{w}_a f_a}{\sum_{s\in\mathcal{S}'} \hat{w}_s f_s}\right| \leq \frac{2(1+\beta)KN\sqrt{N\ln(2K/\delta)}}{\sqrt{2M} - (1+\beta)K(N+1)\sqrt{N\ln(2K/\delta)}}\,.$$

A3.8 Learning Without Resets

Although the analyses in Section 3.5 are tailored to learnability in the sense of Definition 4, they can easily be adapted to hold in the alternate setting in which the learner has access only to a single, unbroken trajectory of states. In this alternate model, the learning algorithm observes a polynomially long prefix of a trajectory of states for training, and then must produce a generative model which results in a distribution over the values of the subsequent T states close to the true distribution.

When learning individual crowd affinity models for each agent in this setting, we again assume that we are presented with a set of samples $\mathcal{M}$, where each instance $\mathcal{I}_m \in \mathcal{M}$ consists of a pair $\langle \mathbf{f}_m, a_m\rangle$. However, instead of assuming that the state distributions $\mathbf{f}_m$ are distributed according to D^f, we now assume that the state and action pairs represent a single trajectory. As previously noted, the majority of the analysis for both the mixture and multiplicative variants of the crowd affinity model does not depend on the particular way in which state distribution vectors are distributed, and thus carries over to this setting as is. Here we briefly discuss the few modifications that are necessary.

The only change required in the analysis of the crowd affinity mixture model relates to handling

the case in which Z_a is small for all a. Previously we argued that when this is the case, the distribution D^f must be concentrated so that for all a, f_a falls within a very small range with high probability. Thus it is not necessary to estimate the parameter α directly, and we can instead learn a single probability for each action that is used regardless of $\mathbf{f}$. A similar argument holds in the no-reset variant. If it is the case that Z_a is small for all a, then it must be the case that for each a, the value of f_a has fallen into the same small range for the entire observed trajectory. A standard uniform convergence argument says that the probability that f_a suddenly changes dramatically is very small, and thus again it is sufficient to learn a single probability for each action that is used regardless of $\mathbf{f}$.

To adapt the analysis of the crowd affinity multiplicative model, it is first necessary to replace Lemma 12. Recall that the purpose of this lemma was to show that when the data set does not contain sufficient samples in which $f_a > 0$ and $f_b > 0$ for a pair of actions a and b, the chance of observing a new state distribution $\mathbf{f}$ with $f_a > 0$ and $f_b > 0$ is small. This argument is actually much more straightforward in the no-reset case. By the definition of the model, it is easy to see that if $f_a > 0$ for some action a at time t in a trajectory, then it must be the case that $f_a > 0$ at all previous points in the trajectory. Thus if $f_a > 0$ on any test instance, then f_a must have been non-negative on *every* training instance, and we do not have to worry about the case in which there is insufficient data to compare the weights of a particular pair of actions.

One additional, possibly more subtle, modification is necessary in the analysis of the multiplicative model to handle the case in which $\chi_{a,b} = \chi_{b,a} = 0$ for all "active" pairs of actions $a, b \in \mathcal{S}'$. This can happen only if agent i has extremely small weights for every action in $\mathcal{S}'$, and had previously been choosing an alternate action that is no longer available, i.e., an action s for which f_s had previously been non-negative but suddenly is not. However, in order for f_s to become 0, it must be the case that agent i himself chooses an alternate action (say, action a) instead of s, which cannot happen since the estimated weight of action a used by the model is 0. Thus this situation can never occur in the no-reset variant.

A4 Additional Proofs from Chapter 4

A4.1 Proof of Theorem 16

The details of the proof follow the general sketch given in Section 4.5. The first step is to show how the adversary can generate a "bad" sequence of gains for any algorithm $\mathcal{A}$. The second step is to show that in order to prove a general lower bound using this sequence generation procedure, it is sufficient to restrict our attention only to algorithms that satisfy a certain set of properties, which we call monotone f-compliant algorithms. In the third step, we break the sequence of gains into "segments" and derive upper and lower bounds on the regret of any f-compliant algorithm in each segment. We next show that it is possible to bound the total number of segments using the fact that the algorithm we are considering has a worst case regret of $\alpha\sqrt{T}$ to the best expert. Finally, we show how to put these pieces together to achieve the main result.

Step 1: Generating the Sequence of Gains

Fix a constant $\alpha > 0$. Figure A.1 shows a procedure that, given an algorithm $\mathcal{A}$, generates a sequence of expert gains $\mathbf{g}$ of length T (for any $T > (150\alpha)^2$) such that $\mathbf{g}$ is a "bad sequence" for $\mathcal{A}$. In this procedure, the variable d_t keeps track of the difference between the gains of the two experts at time t. At each time step, this difference either increases by one or decreases by one, since one expert receives a gain of one and the other zero. The variable $last(d)$ holds the probability that the algorithm assigned to the leading expert the most recent time that the distance between expert gains was d. The variable ϵ_t then represents the difference between the probability that the algorithm assigned to the current best expert at the last time step at which the difference in expert gains was smaller than d_{t-1} and the probability that the algorithm assigns to the best expert for the upcoming time step t. This is used by the sequence generation algorithm to ensure that the best expert will only do well when the algorithm does not have "too much" weight on it. The function f and parameter γ used in the procedure will be defined later in the analysis.

The sequence of gains generated by the procedure in Figure A.1 is specifically designed to fool the algorithm $\mathcal{A}$. In particular, whenever the algorithm updates its weights aggressively, the procedure assigns a positive gain to the second-place expert, inducing mean reversion. On the contrary, when the algorithm updates its weights conservatively, the procedure assigns a positive

Figure A.1 The *GenerateBadSeq* procedure for creating a "bad sequence" of gains.

// Input: An algorithm $\mathcal{A}$, function f, and value γ

Set $t = 1$, $G_{avg,0} = G_{\mathcal{A},0} = d_0 = 0$
while $(G_{avg,t-1} - G_{\mathcal{A},t-1} \leq 0.115\sqrt{T}/\gamma)$ **do**
  $p_{1,t} = \mathcal{A}(\mathbf{g}), p_{2,t} = 1 - \mathcal{A}(\mathbf{g})$
  if $(d_{t-1} = 0)$ **then**
    if $\left(p_{1,t} \leq \frac{1}{2}\right)$ **then**
      $g_{1,t} = 1, g_{2,t} = 0, last(|d_{t-1}|) = p_{1,t}$
    else
      $g_{1,t} = 0, g_{2,t} = 1, last(|d_{t-1}|) = p_{2,t}$
    end if
  else
    $i_t = \text{argmax}_i\, G_{i,t-1}, j_t = \text{argmin}_j\, G_{j,t-1}$
    $last(|d_{t-1}|) = p_{i_t,t}$
    $\epsilon_t = p_{i_t,t} - last(|d_{t-1} - 1|)$
    if $(\epsilon_t \leq f(|d_{t-1}|))$ **then**
      $g_{i_t,t} = 1, g_{j_t,t} = 0$
    else
      $g_{i_t,t} = 0, g_{j_t,t} = 1$
    end if
  end if
  $G_{\mathcal{A},t} = G_{\mathcal{A},t-1} + p_{1,t}g_{1,t} + p_{2,t}g_{2,t}$
  $G_{avg,t} = G_{avg,t-1} + (g_{1,t} + g_{2,t})/2$
  $d_t = d_{t-1} + g_{1,t} - g_{2,t}$
  $t = t + 1$
end while
$g_{1,t} = g_{2,t} = 1/2$ for the rest of the sequence

gain to the best expert, causing added momentum. These competing issues of conservative versus aggressive updates force the algorithm to have "bad" regret to either the best expert or the average on some sequence of gains.

Step 2: Restricting Attention to Monotone f-Compliant Algorithms

We next show that in order to prove a lower bound using the sequence generation procedure defined in Figure A.1, it is sufficient to consider only a specific class of algorithms. In particular, we first show that for any arbitrary algorithm, there exists an *f-compliant* algorithm (to be defined shortly) with equivalent or betters gains for which *GenerateBadSeq* produces the same sequence. Thus we may restrict our attention to only f-compliant algorithms. We then show that any f-compliant

algorithm can be transformed into a *monotone* f-compliant algorithm (defined below) which has performance at least as good, allowing us to further restrict our attention to only monotone f-compliant algorithms.

We begin with the definition of f-compliant. We say that an algorithm $\mathcal{A}$ is *f-compliant* (for a specific function f which will be defined shortly) if at every time t we have (1) $\epsilon_t = f(d_{t-1}) \pm \delta$, for an arbitrarily small δ (for example $\delta = 1/T^2$), and (2) $p_{1,t} = p_{2,t} = 1/2$ if $d_{t-1} = 0$. Since δ can be arbitrarily small, we can think of this requirement as enforcing that ϵ_t be *exactly* equal to $f(d_{t-1})$, and allowing the algorithm to "choose" whether it should be considered larger or smaller. The following lemma implies that given the sequence generation process in Figure A.1, we need only to consider the class of f-compliant algorithms, since for any other algorithm that does not have $\Omega(\sqrt{T})$ regret to the average, there exists an f-compliant algorithm with better gains for which the same sequence is generated.

Lemma 16 *Consider any algorithm $\mathcal{A}$ such that for all $t < T$, $G_{avg,t-1} - G_{\mathcal{A},t-1} \leq 0.115\sqrt{T}/\gamma$, and let* $\mathbf{g} = GenerateBadSeq(\mathcal{A}, f, \gamma)$. *There exists an f-compliant algorithm $\mathcal{A}'$ such that* $GenerateBadSeq(\mathcal{A}', f, \gamma) = \mathbf{g}$ *and at every time $t \leq T$, $g_{\mathcal{A}',t} \geq g_{\mathcal{A},t}$.*

Proof: First consider any time t at which $d_{t-1} = 0$. When this is the case, the procedure will always assign a gain of 1 to the expert with the lower probability. Thus if $\mathcal{A}$ sets $p_{1,t} < p_{2,t}$ or $p_{2,t} < p_{1,t}$, it is possible to achieve a higher gain by setting $p_{1,t} = p_{2,t} = 1/2$ without altering the sequence $\mathbf{g}$ generated by $GenerateBadSeq$.

Suppose $d_{t-1} \neq 0$. We can assume without loss of generality that $d_{t-1} > 0$. Note that when $\epsilon_t \leq f(|d_{t-1}|)$ we have a gain $g_{1,t} = 1$, so maximizing ϵ_t by setting it arbitrarily close to $f(|d_{t-1}|)$ increases the gain without changing $GenerateBadSeq(\mathcal{A}', f, \gamma)$. Similarly, when $\epsilon_t > f(|d_{t-1}|)$ we have $g_{2,t} = 1$, so minimizing ϵ_t by setting it arbitrarily close to $f(|d_{t-1}|)$ maximizes the gain of $\mathcal{A}$ without changing $GenerateBadSeq(\mathcal{A}', f, \gamma)$. In both cases, letting ϵ_t approach $f(|d_{t-1}|)$ is better for the algorithm and thus the modified algorithm $\mathcal{A}'$ will always have a higher payoff on $GenerateBadSeq(\mathcal{A}', f, \gamma)$. ■

Given an f-compliant algorithm, we can write its probabilities as a function of the difference between expert gains. In particular, we define a function $F(d) = 1/2 + \sum_{i=1}^{|d|} f(i)$, where $F(0) = 1/2$. It is easy to verify that an algorithm $\mathcal{A}$ that sets the probability of the best expert at time

t to $F(d_{t-1})$ is an f-compliant algorithm. Furthermore, as δ approaches 0, *every* f-compliant algorithm will assign expert weights arbitrarily close to these weights. It is convenient to think of the algorithm weights in this way for the next steps of the analysis.

We are now ready to define the function f used in sequence generation. Let

$$f(d) = \frac{2^{m(d)-1}}{\gamma\sqrt{T}} \quad \text{where} \quad m(d) = \left\lceil \frac{16\alpha}{\sqrt{T}}|d| \right\rceil .$$

It then follows that

$$F(d) = \frac{1}{2} + \sum_{i=1}^{|d|} \frac{2^{m(i)-1}}{\gamma\sqrt{T}} \leq \frac{1}{2} + \sum_{j=1}^{m(d)} \frac{2^{j-1}}{\gamma\sqrt{T}} \left(\frac{\sqrt{T}}{16\alpha}\right) \leq \frac{1}{2} + \frac{2^{m(d)}}{16\gamma\alpha} . \tag{1}$$

We next define the (possibly noncontiguous) m *segment* $\mathcal{T}_m$ to be the set of all times t for which $m(d_t) = m$. More explicitly,

$$\mathcal{T}_m = \{t : (m-1)(\sqrt{T}/(16\alpha)) \leq |d_t| < m(\sqrt{T}/(16\alpha))\} .$$

Based on this, we define a *monotone* f-compliant algorithm to be an f-compliant algorithm $\mathcal{A}$ such that when $GenerateBadSeq$ is applied to $\mathcal{A}$ it is the case that for all m and m', for all $t \in \mathcal{T}_m$ and $t' \in \mathcal{T}_{m'}$, if $m < m'$ then $t < t'$. In other words, an f-compliant algorithm is monotone if every m segment consists of a contiguous set of time steps. The following observation is useful in simplifying the proof, allowing us to further restrict our attention to the class of monotone f-compliant algorithms. It says that a lower bound on the performance of monotone algorithms will imply the general lower bound.

Lemma 17 *Consider any non-monotone f-compliant algorithm $\mathcal{A}$, and let* $\mathbf{g} = GenerateBadSeq(\mathcal{A}, f, \gamma)$. *There exists a monotone f-compliant algorithm $\mathcal{A}'$ with* $\mathbf{g}' = GenerateBadSeq(\mathcal{A}', f, \gamma)$ *such that*

$$\sum_{t=1}^{T} g'_{\mathcal{A}',t} > \sum_{t=1}^{T} g_{\mathcal{A},t}.$$

Proof: If $\mathcal{A}$ is not monotone, then there must be some time step t and some distance $d > 0$ such that $m(d+2) = m(d+1) + 1$ and $|d_t| = d$, $|d_{t+1}| = d+1$, $|d_{t+2}| = d+2$, and $|d_{t+3}| = d+1$.

Here the first crossover into the $m(d+2)$ segment occurs at time $t+2$, and we cross back into the $m(d+1)$ segment at time $t+3$. Since $\mathcal{A}$ is f-compliant,

$$g_{\mathcal{A},t+1} + g_{\mathcal{A},t+2} + g_{\mathcal{A},t+3} = F(d) + F(d+1) + (1 - F(d+2)) = F(d) + 1 - f(d+2) \; .$$

Now, consider a modified f-compliant algorithm $\mathcal{A}'$ that is the same as $\mathcal{A}$ everywhere except it chooses to have the weight it places on the leading expert at time $t+2$ treated as arbitrarily close to but *greater than* $1/2 + F(d+1)$ instead of arbitrarily close to but *less than* $1/2 + F(d+1)$, and sets the weight of this expert at time $t+3$ arbitrarily close to but less than $1/2 + F(d)$. This has the effect of modifying the sequence of distances so that $|d_{t+2}| = d$; the rest of the sequence remains the same. On this modified sequence,

$$g'_{\mathcal{A}',t+1} + g'_{\mathcal{A}',t+2} + g'_{\mathcal{A}',t+3} = F(d) + (1 - F(d+1)) + F(d) = F(d) + 1 - f(d+1) \; .$$

Since $m(d+2) > m(d+1)$, it must be the case that $f(d+2) > f(d+1)$ and the total gain of $\mathcal{A}'$ is strictly higher than the total gain of $\mathcal{A}$.

If $\mathcal{A}'$ is not monotone, this transformation process can be repeated until a monotone f-compliant algorithm is found. Each time, the gain of the algorithm will strictly increase, yielding the result. ■

The above lemma shows how we can change an f-compliant algorithm into a monotone f-compliant algorithm whose performance is at least as good. Therefore, we can consider only monotone algorithms.

Step 3: Bounding the Algorithm's Regret from Above and Below in Each Segment

Now that we have established that it suffices to consider only the performance of f-compliant algorithms on the sequence of gains generated by the procedure in Figure A.1, we are ready to introduce the notion of *matched times* and *unmatched times*. We define a pair of matched times as two times t_1 and t_2 such that the difference between the cumulative gains the two experts changes from d to $d+1$ by time t_1 and stays at least as high as $d+1$ until changing from $d+1$ back to d at time t_2. More formally, for some difference d, $d_{t_1-1} = d_{t_2} = d$, and for all t such that $t_1 \leq t < t_2$, $d_t > d$. Clearly each pair of matched times consists of one time step in which the gain of one

expert is 1 and the other 0 while at the other time step the reverse holds. We refer to any time at which one expert has gain 1 while the other has gain 0 that is *not* part of a pair of matched times as an unmatched time. If at any time t we have $d_t = d$, then there must have been d unmatched times at some point before time t.

We denote by $\mathcal{M}_m$ and $\mathcal{UM}_m$ the matched and unmatched times in the m segment $\mathcal{T}_m$, respectively. These concepts will become important due to the fact that an algorithm will lose with respect to the average for every pair of matched times, but will gain with respect to the average on every unmatched time.

The following lemma quantifies the regret of the algorithm to the best expert and the average of all experts for each pair of matched times.

Lemma 18 *For any f-compliant algorithm $\mathcal{A}$ and any pair of matched times t_1 and t_2 in the m segment, the gain of the algorithm from times t_1 and t_2 (i.e., $g_{\mathcal{A},t_1} + g_{\mathcal{A},t_2}$) is $1 - 2^{m-1}/(\gamma\sqrt{T})$, while the gain of the average and the best expert is 1.*

Proof: Let $d = d_{t_1} - 1$. Without loss of generality assume that the leading expert is expert 1, i.e., $d \geq 0$. The gain of the algorithm at time t_1 is $p_{1,t_1} = F(d)$, while the gain at t_2 is $p_{2,t_2} = 1 - p_{1,t_2} = 1 - F(d+1) = 1 - (F(d) + f(d))$. Thus the algorithm has a total gain of $1 - f(d) = 1 - 2^{m-1}/(\gamma\sqrt{T})$ for these time steps. ■

On the other hand, the following lemma provides an upper bound on the gain of the algorithm over the average expert from the unmatched times only.

Lemma 19 *The gain of any f-compliant algorithm $\mathcal{A}$ in only the unmatched times in the m segment of the algorithm is at most $2^m\sqrt{T}/(256\gamma\alpha^2)$ larger than the gain of the average expert in the unmatched times in segment m, i.e.,*

$$\sum_{t \in \mathcal{UM}_m} \left(g_{\mathcal{A},t} - \frac{1}{2} \right) \leq \frac{2^m\sqrt{T}}{256\gamma\alpha^2} \,.$$

Proof: Since the leading expert does not change in the unmatched times (in retrospect), we can assume w.l.o.g. that it is expert 1. From (1), it follows that

$$\sum_{t \in \mathcal{UM}_m} g_{\mathcal{A},t} - 1/2 \leq \sum_{i=0}^{\frac{\sqrt{T}}{16\alpha}-1} \left(F(d+i) - \frac{1}{2} \right) \leq \frac{2^m}{16\gamma\alpha} \frac{\sqrt{T}}{16\alpha} \leq \frac{2^m\sqrt{T}}{256\gamma\alpha^2} \,.$$

■

Combining Lemmas 18 and 19, we can compute the number of matched times needed in the m segment in order for the loss of the algorithm to the average from matched times to cancel the gain of the algorithm over the average from unmatched times.

Lemma 20 *For any fixed integer x, if there are at least $T/(128\alpha^2) + x$ pairs of matched times in the m segment, then the gain of any f-compliant algorithm $\mathcal{A}$ in the m segment is bounded by the gain of the average expert in the m segment minus $x2^{m-1}/(\gamma\sqrt{T})$, i.e.,*

$$\sum_{t \in \mathcal{T}_m} g_{\mathcal{A},t} \leq \sum_{t \in \mathcal{T}_m} \frac{1}{2} - \frac{2^{m-1}x}{\gamma\sqrt{T}} \ .$$

Proof: From Lemma 19, $\mathcal{A}$ can not gain more than $2^m\sqrt{T}/(256\alpha^2\gamma)$ over the average in the m segment. From Lemma 18, the loss of $\mathcal{A}$ with respect to the average for each pair of matched times is $2^{m-1}/(\gamma\sqrt{T})$. Since there are at least $T/(128\alpha^2) + x$ pairs of matched times, the *total* amount the algorithm loses to the average in the m segment is at least $2^{m-1}x/(\gamma\sqrt{T})$. ■

Step 4: Bounding the Number of Segments

The next lemma bounds the number of segments in the sequence using the fact that $\mathcal{A}$ is an $\alpha\sqrt{T}$-regret algorithm.

Lemma 21 *For any f-compliant algorithm $\mathcal{A}$ such that $R_{best,\mathcal{A},T} < \alpha\sqrt{T}$ and for $\gamma = 2^{48\alpha^2}/\alpha$, there are at most $48\alpha^2$ segments in $\mathbf{g} = GenerateBadSeq(\mathcal{A}, f, \gamma)$.*

Proof: Once again we assume that leading expert is expert 1. Setting $\gamma = 2^{48\alpha^2}/\alpha$ in (1), ensures that $F(d)$ is bounded by $2/3$ as long as m remains below $48\alpha^2$. Thus $F(d)$ is bounded by $2/3$ for all unmatched times until we reach segment $48\alpha^2$. This implies that if the sequence reaches segment $48\alpha^2$, then the regret with respect to the best expert will be at least $48\alpha^2\sqrt{T}/(16\alpha)(1/3) = \alpha\sqrt{T}$ which contradicts the fact that $\mathcal{A}$ is a $\alpha\sqrt{T}$-regret algorithm, so it cannot be the case that the sequence has $48\alpha^2$ or more segments. ■

Step 5: Putting the Pieces Together

We are now ready to prove the main lower bound theorem.

First, consider the case in which the main `while` loop of $GenerateBadSeq(\mathcal{A}, f, \gamma)$ terminates before time T. It must be the case that $G_{avg,t-1} - G_{\mathcal{A},t-1} > 0.115\sqrt{T}/\gamma = \Omega(\sqrt{T})$ and there is nothing more to prove.

Throughout the rest of the proof, assume that the main `while` loop is never exited while generating the sequence $\mathbf{g}$. From Lemma 20 we know that if there are at least $T/(128\alpha^2)$ pairs of matched times in the m segment, then the loss to the average from these times will cancel the gain from unmatched times in this segment. By Lemma 21 there are at most $48\alpha^2$ segments. If the algorithm has *exactly* $T/(128\alpha^2)$ pairs of matched times at each segment, it will have at most a total of $T/(128\alpha^2)(48\alpha^2) = (3/8)T$ pairs of matched times and will cancel all of its gain over the average from the unmatched times in all segments. Note that there are at most $48\alpha^2\sqrt{T}/(16\alpha) = 3\alpha\sqrt{T}$ unmatched times. Since we have chosen T such that $\alpha < \sqrt{T}/150$, we can bound this by $0.02T$. This implies that there are at least $0.49T$ pairs of matched times. We define the following quantity for algorithm $\mathcal{A}$: $x_m = |\mathcal{M}_m|/2 - T/(128\alpha^2)$. We have that

$$\sum_{m=1}^{48\alpha^2} x_m = \left(\sum_{m=1}^{48\alpha^2} \frac{|\mathcal{M}_m|}{2}\right) - \frac{3T}{8} \geq 0.49T - (3/8)T = 0.115T\ .$$

Let m^* be the first segment for which we have $\sum_{i=1}^{m^*} x_i \geq 0.115T$. Since we consider only monotone algorithms we know that by that time no segments larger than m^* have been visited. For every k, $1 \leq k \leq m^*$, we have $z_k = \sum_{i=k}^{m^*} x_i > 0$ (otherwise m^* would not be the first segment). Note that we can bound the regret to the average from below as follows,

$$\begin{aligned}
\sum_{i=1}^{m^*} x_i \frac{2^{i-1}}{\gamma\sqrt{T}} &= \frac{1}{\gamma\sqrt{T}} x_1 + \frac{1}{\gamma\sqrt{T}} \sum_{i=2}^{m^*} x_i \left(1 + \sum_{j=1}^{i-1} 2^{j-1}\right) \\
&= \frac{1}{\gamma\sqrt{T}} \sum_{i=1}^{m^*} x_i + \frac{1}{\gamma\sqrt{T}} \sum_{i=2}^{m^*} \sum_{j=2}^{i} 2^{j-2} x_i \\
&= \frac{1}{\gamma\sqrt{T}} z_1 + \frac{1}{\gamma\sqrt{T}} \sum_{j=2}^{m^*} 2^{j-2} z_j \geq \frac{0.115T}{\gamma\sqrt{T}} = \frac{0.115\sqrt{T}}{\gamma}\ .
\end{aligned}$$

This shows that the regret to the average must be at least $0.115\sqrt{T}/\gamma = \beta\sqrt{T}$ where $\beta = 0.115\alpha/2^{48\alpha^2}$, yielding the first result of the theorem.

Finally, for any $\alpha' \leq 1/10$, let $\alpha = \alpha'\sqrt{\log T} \leq \sqrt{\log T}/10$. From the previous result, this

implies that if the regret to the best expert is bounded by $\alpha'\sqrt{T\log T} = \alpha\sqrt{T}$, then the regret to the average must be at least $(0.115\alpha/2^{(48/100)\log T})\sqrt{T} = 0.115\alpha T^{1/2-48/100} = \Omega(T^{1/50})$. This proves the second part of the theorem.

A5 Additional Proofs from Chapter 5

A5.1 Proof of Theorem 18

Let P be a partial order over $\{1,\ldots,n\}$. Recall that a linear (or total) order T is a *linear extension* of P if whenever $x \leq y$ in P it also holds that $x \leq y$ in T. We denote by $\mathcal{N}(P)$ the number of linear extensions of P.

Recall that (i,j) is a *covering pair* of P if $i \leq j$ in P and there does not exist $\ell \neq i,j$ such that $i \leq \ell \leq j$. Let $\{(i_1,j_1), (i_2,j_2), \ldots, (i_k,j_k)\}$ be a set of covering pairs of P. Note that covering pairs of a partially ordered set with n elements can be easily obtained in polynomial time, and that their number is less than n^2.

We will show that we can design a sequence of trades that, given a list of covering pairs for P, provides $\mathcal{N}(P)$ through a simple function of market prices.

We consider a pair betting market over n candidates. We construct a sequence of k trading periods, and denote by $q^t_{i,j}$ and $p^t_{i,j}$ respectively the outstanding quantity of security $\langle i > j\rangle$ and its instantaneous price at the end of period t. At the beginning of the market, $q^0_{i,j} = 0$ for any i and j. At each period t, $0 < t \leq k$, $b\ln n!$ shares of security $\langle i_t > j_t\rangle$ are purchased.

Let

$$N_t(i,j) = \sum_{\sigma\in\Omega:\sigma(i)<\sigma(j)} \prod_{i',j':\sigma(i')<\sigma(j')} \mathrm{e}^{q^t_{i',j'}/b},$$

and

$$D_t = \sum_{\sigma\in\Omega} \prod_{i',j':\sigma(i')<\sigma(j')} \mathrm{e}^{q^t_{i',j'}/b}.$$

Note that according to Equation 5.9, $p^t_{i_t,j_t} = N_t(i_t,j_t)/D_t$.

For the first period, as only the security $\langle i_1 > j_1\rangle$ is purchased, we get

$$D_1 = \sum_{\sigma\in\Omega:\sigma(i_1)<\sigma(j_1)} n! + \sum_{\sigma:\sigma(i_1)>\sigma(j_1)} 1 = \frac{(n!)^2 + n!}{2}.$$

We now show that D_k can be calculated inductively from D_1 using successive prices given by the market. During period t, $b \ln n!$ shares of $\langle i_t > j_t \rangle$ are purchased. Note also that the securities purchased are different at each period, so that $q^s_{i_t,j_t} = 0$ if $s < t$ and $q^s_{i_t,j_t} = b \ln n!$ if $s \geq t$. We have

$$N_t(i_t, j_t) = N_{t-1}(i_t, j_t) \mathrm{e}^{b \ln(n!)/b} = n! N_{t-1}(i_t, j_t) \; .$$

Hence,

$$\frac{p^t_{i_t,j_t}}{p^{t-1}_{i_t,j_t}} = \frac{N_t(i_t, j_t)/D_t}{N_{t-1}(i_t, j_t)/D_{t-1}} = \frac{n! D_{t-1}}{D_t} \; ,$$

and therefore,

$$D_k = (n!)^{k-1} \left(\prod_{\ell=2}^{k} \frac{p^{\ell-1}_{i_\ell,j_\ell}}{p^{\ell}_{i_\ell,j_\ell}} \right) D_1 \; .$$

So D_k can be computed in polynomial time in n given the prices.

Alternately, since the cost function at the end of period k can be written as $C(Q) = b \log D_k$, D_k can also be computed efficiently from the cost function in period k.

We finally show that given D_k, we can compute $\mathcal{N}(P)$ in polynomial time. Note that at the end of the k trading periods, the securities purchased correspond to the covering pairs of P, such that $\mathrm{e}^{q^k_{i,j}/b} = n!$ if (i, j) is a covering pair of P and $\mathrm{e}^{q^k_{i,j}/b} = 1$ otherwise. Consequently, for a permutation σ that satisfies the partial order P, meaning that $\sigma(i) \leq \sigma(j)$ whenever $i \leq j$ in P, we have

$$\prod_{i',j':\sigma(i')<\sigma(j')} \mathrm{e}^{q^k_{i',j'}/b} = (n!)^k \; .$$

On the other hand, if a permutation σ does not satisfy P, it does not satisfy at least one covering pair, meaning that there is a covering pair of P, (i, j), such that $\sigma(i) > \sigma(j)$, so that

$$\prod_{i',j':\sigma(i')<\sigma(j')} \mathrm{e}^{q^k_{i',j'}/b} \leq (n!)^{k-1} \; .$$

Since the total number of permutations is $n!$, the total sum of *all* terms in the sum D_k corresponding to permutations that do not satisfy the partial ordering P is less than or equal to $n!(n!)^{k-1} = (n!)^k$, and is strictly less than $(n!)^k$ unless the number of linear extensions is 0, while the total sum of all the terms corresponding to permutations that do satisfy P is $\mathcal{N}(P)(n!)^k$. Thus $\mathcal{N}(P) = \lfloor D_k/(n!)^k \rfloor$.

We know that computing the number of linear extensions of a partial ordering is #P-hard. Therefore, both computing the prices and computing the value of the cost function in pair betting are #P-hard.

A5.2 Proof of Theorem 19

Suppose we are given a 2-CNF (Conjunctive Normal Form) formula

$$(X_{i_1} \vee X_{j_1}) \wedge (X_{i_2} \vee X_{j_2}) \wedge \cdots \wedge (X_{i_k} \vee X_{j_k}) \tag{2}$$

with k clauses, where each clause is a disjunction of two literals (i.e. events and their negations). Assume any redundant terms have been removed.

The structure of the proof is similar to that of the pair betting case. We consider a Boolean betting market with N events, and show how to construct a sequence of trades that provides, through prices or the value of the cost function, the number of satisfiable assignments for the 2-CNF formula.

We create k trading periods. At period t, a quantity $b\ln(2^N)$ of the security $\langle X_{i_t} \vee X_{j_t} \rangle$ is purchased. We denote by $p^t_{i,j}$ and $q^t_{i,j}$ respectively the price and outstanding quantities of the security $\langle X_i \vee X_j \rangle$ at the end of period t. Suppose the market starts with 0 share of every security. Then $q^s_{i_t,j_t} = 0$ if $s < t$ and $q^s_{i_t,j_t} = b\ln(2^N)$ if $s \geq t$. Let

$$N_t(i,j) = \sum_{\omega\in\Omega:\omega\in(X_i\vee X_j)} \prod_{1\leq i'<j'\leq 2N:\omega\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q^t_{i',j'}/b},$$

and

$$D_t = \sum_{\omega\in\Omega} \prod_{1\leq i'<j'\leq 2N:\omega\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q^t_{i',j'}/b}.$$

Thus, $p^t_{i,j} = N_t(i_t, j_t)/D_t$.

Since only one security $\langle X_{i_1} \vee X_{j_1} \rangle$ has been purchased in period 1, we get

$$D_1 = \sum_{\omega\in\Omega:\omega\in(X_{i_1}\vee X_{j_1})} 2^N + \sum_{\omega\in\Omega:\omega\notin(X_{i_1}\vee X_{j_1})} 1 = 3\cdot 2^{2N-2} + 2^{N-2}.$$

We then show that D_k can be calculated inductively from D_1. As the only security purchased

in period t is $(X_{i_t} \vee X_{j_t})$ in quantity $b\ln(2^N)$, we obtain

$$N_t(i_t, j_t) = N_{t-1}(i_t, j_t)\mathrm{e}^{b\ln(2^N)/b} = N_{t-1}(i_t, j_t)2^N \ .$$

Therefore,

$$\frac{p^t_{i_t,j_t}}{p^{t-1}_{i_t,j_t}} = \frac{N_t(i_t, j_t)/D_t}{N_{t-1}(i_t, j_t)/D_{t-1}} = \frac{2^N D_{t-1}}{D_t},$$

and we get

$$D_k = (2^N)^{k-1}\left(\prod_{\ell=2}^{k}\frac{p^{\ell-1}_{i_\ell,j_\ell}}{p^{\ell}_{i_\ell,j_\ell}}\right)D_1 \ .$$

In addition, since the cost function at the end of period k can be expressed as

$$C(Q) = b\log D_k \ ,$$

D_k can also be computed efficiently from the cost function in period k.

We now show that we can deduce from D_k the number of satisfiable assignments for the 2-CNF formula (Equation 2). Indeed, each term in the sum

$$\sum_{\omega\in\Omega}\prod_{1\le i'<j'\le 2N:\omega\in(X_{i'}\vee X_{j'})}\mathrm{e}^{q^k_{i',j'}/b}$$

that corresponds to an outcome ω that satisfies the formula is exactly 2^{kN}, as exactly k terms in the product are 2^N and the rest are 1. On the contrary, each term in the sum that corresponds to an outcome ω that does *not* satisfy the 2-CNF formula will be at most $2^{(k-1)N}$ since at most $k-1$ terms in the product will be 2^N and the rest will be 1. Since the total number of outcomes is 2^N, the total sum of *all* terms corresponding to outcomes that do not satisfy Equation 2 is less than or equal to $2^N(2^{(k-1)N}) = 2^{kN}$, and is strictly less than 2^{kN} unless the number of satisfying assignments is 0. Thus the number of satisfying assignments is $\lfloor D_k/2^{kN}\rfloor$.

We know that computing the number of satisfiable assignments of a 2-CNF formula is #P-hard. We have shown how to compute it in polynomial time using prices or the value of the cost function in a Boolean betting market of N events. Therefore, both computing prices and computing the value of the cost function in a Boolean betting market is #P-hard.

References

[1] P. Abbeel, D. Koller, and A. Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7:1743–1788, 2006.

[2] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009.

[3] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Tauman Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *Proceedings of the 17th International World Wide Web Conference*, 2008.

[4] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[5] P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64:48–75, 2002.

[6] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 619–626, 2001.

[7] H. Balakrishnan, I. Hwang, and C. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 4874–4879, 2004.

[8] M.-F. Balcan, A. Blum, J. Hartline, and Yishay Mansour. Reducing mechanism design to algorithm design via machine learning. *Journal of Computer and System Sciences*, 74: 1245–1270, 2008.

[9] M.-F. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

[10] P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[11] P. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2002.

[12] J. Baxter. Learning internal representations. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, 1995.

[13] R. Bell and Y. Koren. Lessons from the Netflix prize challenge. *SIGKDD Explorations*, 9 (2), 2007.

[14] R. Bell, Y. Koren, and C. Volinsky. Chasing $1,000,000: How we won the Netflix progress prize. *ASA Statistical and Computing Graphics Newsletter*, 18(2), 2007.

[15] S. Ben-David. Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, 2003.

[16] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19*, 2006.

[17] J. Berg and T. Rietz. Accuracy and forecast standard error of prediction markets. Technical report, University of Iowa, College of Business Administration, 2002.

[18] N. Berger, C. Borgs, J. T. Chayes, and A. Saberi. On the spread of viruses on the Internet. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithm*, 2005.

[19] J. Blitzer. *Domain Adaptation of Natural Language Processing Systems*. PhD thesis, University of Pennsylvania, 2007.

[20] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 20*, 2007.

[21] J. Blitzer, M. Dredze, and F. Pereira. Biographies, Bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

[22] G. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, 1991.

[23] G. W. Brown. Iterative solutions of games by fictitious play. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, 1951.

[24] T. Bylander. Learning noisy linear threshold functions. Technical Report, 1998.

[25] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[26] N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2/3):321–352, 2007.

[27] C. Chelba and A. Acero. Empirical methods in natural language processing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[28] Y. Chen and D. M. Pennock. A utility framework for bounded-loss market makers. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 49–56, 2007.

[29] Y. Chen, L. Fortnow, E. V. Nikolova, and D. M. Pennock. Betting on permutations. In *Proceedings of the Eighth ACM Conference on Electronic Commerce*, 2007.

[30] Y. Chen, D. Reeves, D. M. Pennock, R. Hanson, L. Fortnow, and R. Gonen. Bluffing and strategic reticence in prediction markets. In *Proceedings of the 3rd International Workshop on Internet and Network Economics*, 2007.

[31] Y. Chen, L. Fortnow, N. Lambert, D. M. Pennock, and J. Wortman. Complexity of combinatorial market makers. In *Proceedings of the Ninth ACM Conference on Electronic Commerce*, 2008.

[32] Y. Chen, S. Goel, and D. M. Pennock. Pricing combinatorial markets for tournaments. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, 2008.

[33] N. A. Christakis and J. H. Fowler. The spread of obesity in a large social network over 32 years. *New England Journal of Medicine*, 357:370–379, 2007.

[34] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[35] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proceedings of the 38th IEEE Annual Symposium on Foundations of Computer Science*, 1997.

[36] J. S. Coleman, E. Katz, and H. Menzel. *Medical innovation: A Diffusion Study*. Bobbs-Merrill, Indianapolis, MN, 1966.

[37] T. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–19, 1991.

[38] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 1991.

[39] K. Crammer, M. Kearns, and J. Wortman. Learning from data of variable quality. In *Advances in Neural Information Processing Systems 18*, 2005.

[40] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9:1757–1774, 2008. Preliminary version appeared in Advances in Neural Information Processing Systems 19.

[41] S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems 18*, 2005.

[42] S. Dasgupta. The sample complexity of learning fixed-structure Bayesian networks. *Machine Learning*, 29(2–3):165–180, 1997.

[43] A. De Vany. *Hollywood Economics: How Extreme Uncertainty Shapes the Film Industry*. Routledge, London, 2004.

[44] O. Dekel and O. Shamir. Good learners for evil teachers. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[45] O. Dekel and O. Shamir. Vox populi: Collecting high-quality labels from a crowd. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

[46] P. Dodds and D. J. Watts. A generalized model of social and biological contagion. *Journal of Theoretical Biology*, 232:587–604, 2005.

[47] P. Dodds, R. Muhamad, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301:828–829, August 2003.

[48] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[49] M. Drehmann, J. Oechssler, and A. Roider. Herding and contrarian behavior in financial markets: An Internet experiment. *American Economic Review*, 95(5):1403–1426, 2005.

[50] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[51] E. Even-Dar, M. Kearns, and J. Wortman. Sponsored search with contexts. In *Proceedings of the 3rd International Workshop on Internet and Network Economics*, 2007.

[52] E. Even-Dar, M. Kearns, Y. Mansour, and J. Wortman. Regret to the best versus regret to the average. *Machine Learning Journal*, 72(1–2):21–37, 2008. Preliminary version appeared in the Proceedings of the Twentieth Annual Conference on Learning Theory.

[53] L. Fortnow, J. Kilian, D. M. Pennock, and M. Wellman. Betting Boolean-style: A framework for trading securities based on logical formulas. *Decision Support Systems*, 39(1):87–104, 2004.

[54] D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29:7–35, 1999.

[55] Y. Freund. Predicting a binary sequence almost as well as the optimal biased coin. *Information and Computation*, 182(2):73–94, 2003.

[56] Y. Freund and R. 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.

[57] Z. Ghahramani. Learning dynamic Bayesian networks. In C.L. Giles and M. Gori, editors, *Adaptive Processing of Sequences and Data Structures*. Springer-Verlag, Berlin, 1998.

[58] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 61:1420–1443, 1978.

[59] S. J. Grossman. An introduction to the theory of rational expectations under asymmetric information. *Review of Economic Studies*, 48(4):541–559, 1981.

[60] D. Gruhl, R. V. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. *SIGKDD Explorations*, 6(2):43–52, 2004.

[61] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, 2001.

[62] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1): 105–119, 2003.

[63] R. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15, 2007.

[64] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[65] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[66] P. Hedstrom. Rational imitation. In P. Hedstrom and R. Swedberg, editors, *Social Mechanisms: An Analytical Approach to Social Theory*. Cambridge University Press, 1998.

[67] D. Helmbold and M. Warmuth. Learning permutations with exponential weights. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 469–483, 2007.

[68] D. Helmbold, R. Schapire, Y. Singer, and M. Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998.

[69] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[70] J. Hu and M. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

[71] J. Jansen and T. Mullen. Sponsored search: An overview of the concept, history, and technology. *International Journal of Electronic Business*, 6(2):114–131, 2008.

[72] A. Kalai and S. Vempala. Efficient algorithms for on-line optimization. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[73] B. Kalantari and L. Khachiyan. On the complexity of nonnegative-matrix scaling. *Linear Algebra and its applications*, 240:87–103, 1996.

[74] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497, 1994.

[75] M. Kearns and S. Singh. Finite-sample rates of convergence for Q-learning and indirect methods. In *Advances in Neural Information Processing Systems 11*, 1998.

[76] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[77] M. Kearns and J. Wortman. Learning from collective behavior. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

[78] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1994.

[79] M. Kearns, R. Schapire, and L. Sellie. Towards efficient agnostic learning. *Machine Learning*, 17:115–141, 1994.

[80] M. Kearns, S. Suri, and N. Montfort. A behavioral study of the coloring problem on human subject networks. *Science*, 313(5788):824–827, 2006.

[81] M. Kearns, J. Tan, and J. Wortman. Privacy-preserving belief propagation and sampling. In *Advances in Neural Information Processing Systems 20*, 2007.

[82] M. Kearns, J. Tan, and J. Wortman. Network-faithful secure computation. Working Draft, 2008.

[83] M. Kearns, J. S. Judd, J. Tan, and J. Wortman. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences*, 106(5):1347–1352, 2009.

[84] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence in a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[85] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, 2005.

[86] J. Kleinberg. Cascading behavior in networks: Algorithmic and economic issues. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[87] V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

[88] V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. *High Dimensional Probability*, II:443–459, 2000.

[89] A. Krause and E. Horvitz. A utility-theoretic approach to privacy and personalization. In *Proceedings of the 23rd Conference on Artificial Intelligence*, 2008.

[90] N. Lambert, J. Langford, J. Wortman, Y. Chen, D. Reeves, Y. Shoham, and D. M. Pennock. Self-financed wagering mechanisms for forecasting. In *Proceedings of the Ninth ACM Conference on Electronic Commerce*, 2008.

[91] J. Ledyard, R. Hanson, and T. Ishikida. An experimental test of combinatorial information markets. *Journal of Economic Behavior and Organization*, 69:182–189, 2009.

[92] C. Legetter and P. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, 9:171–185, 1995.

[93] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International World Wide Web Conference*, 2008.

[94] J. Leskovec, L. Adamic, and B. Huberman. The dynamics of viral marketing. In *Proceedings of the Seventh ACM Conference on Electronic Commerce*, 2006.

[95] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[96] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *Proceedings of the SIAM International Conference on Data Mining*, 2007.

[97] N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000.

[98] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[99] A. Martinez. Recognition of partially occluded and/or imprecisely localized faces using a probabilistic approach. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[100] A. Mas-Collel, W. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[101] A. Maurer. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6:967–994, 2005.

[102] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989.

[103] J. F. Muth. Rational expectations and the theory of price movements. *Econometrica*, 29(6): 315–335, 1961.

[104] N. Nisan. Algorithms for selfish agents. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, 1999.

[105] D. M. Pennock and R. Sami. Computational aspects of prediction markets. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[106] D. M. Pennock, C. L. Giles, and F. A. Nielsen. The real power of artificial markets. *Science*, 291:987–988, 2001.

[107] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[108] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[109] R. Roll. Orange juice and weather. *American Economic Review*, 74(5):861–880, 1984.

[110] B. Ryan and N. C. Gross. The diffusion of hybrid seed corn in two Iowa communities. *Rural Sociology*, 8:15–24, 1943.

[111] M. Salganik and D. J. Watts. Social influence, manipulation, and self-fulfilling prophecies in cultural markets. Preprint, 2007.

[112] M. Salganik, P. Dodds, and D. J. Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 331(5762):854–856, 2006.

[113] T. Schelling. *Micromotives and Macrobehavior*. Norton, New York, NY, 1978.

[114] C. R. Shipan and C. Volden. Bottom-up federalism: The diffusion of antismoking policies from U.S. cities to states. *American Journal of Political Science*, 50(4):825–843, 2006.

[115] Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007.

[116] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[117] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

[118] N. Srebro, N. Alon, and T. Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *Advances in Neural Information Processing Systems 17*, 2004.

[119] F. Stokes Berry and W. D. Berry. State lottery adoptions as policy innovations: An event history analysis. *The American Political Science Review*, 84(2):395–415, 1990.

[120] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International World Wide Web Conference*, 2004.

[121] R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, 1998.

[122] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20 (5):865–877, 1991.

[123] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8: 189–201, 1979.

[124] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3):410–421, 1979.

[125] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[126] H. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

[127] H. Varian. Mechanism design for computerized agents. In *Proceedings of the USENIX Workshop on Electronic Commerce*, 1995.

[128] V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998.

[129] T. Vu, R. Powers, and Y. Shoham. Learning in games with more than two players. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006.

[130] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[131] I. Welch. Herding among security analysts. *Journal of Financial Economics*, 58:369–396, 2000.

[132] J. Wortman. Viral marketing and the diffusion of trends on social networks. Technical Report MS-CIS-08-19, University of Pennsylvania Department of Computer and Information Science, 2008.

[133] P. Wu and T. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.