Using Regret Estimation to Solve Games Compactly

by

Dustin Morrill

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Game theoretic solution concepts, such as Nash equilibrium strategies that are optimal against worst case opponents, provide guidance in finding desirable autonomous agent behaviour. In particular, we wish to approximate solutions to complex, dynamic tasks, such as negotiation or bidding in auctions. Computational game theory investigates effective methods for computing such strategies. Solving human-scale games, however, is currently an intractable problem.

Counterfactual Regret Minimization (CFR) [43], is a regret-minimizing, online learning algorithm that dominates the **Annual Computer Poker Competition (ACPC)** and lends itself readily to various sampling and abstraction techniques. Abstract games are created to mirror the strategic elements of an original game in a more compact representation. The abstract game can be solved and the abstract game solution can be translated back into the full game.

But crafting an abstract game requires domain-specific knowledge, and an abstraction can interact with the game solving process in unintuitive and harmful ways. For example, abstracting a game can create pathologies where solutions to more granular abstractions can be more exploitable against a worst-case opponent in the full game than those derived from simpler abstractions [42]. An abstraction that could be dynamically changed and informed by the solution process could potentially produce better solutions more consistently.

We suggest that such abstractions can be largely subsumed by a regressor on game features that estimates regret during CFR. Replacing abstraction with a regressor allows the memory required to approximate a solution to a game to be proportional to the complexity of the regressor rather than the size of the game itself. Furthermore, the regressor essentially becomes a tunable, compact, and dynamic abstraction of

the game that is informed by and adapts to the particular solution being computed. These properties will allow this technique to scale to previously intractable domains. We call this new algorithm **Regression CFR (RCFR)**.

In addition to showing that this approach is theoretically and practically sound, we improve RCFR by combining it with regret-matching$^+$ [37]. Experiments involving two small poker games show that RCFR and its extension, RCFR$^+$, show that it can approximately solve games with regressors that are drastically less complex than the game itself. In comparisons with traditional static abstractions of similar complexity, RCFR variants tend to produce less exploitable strategies.

Preface

Some of the research conducted for this thesis was a collaboration with PhD student Kevin Waugh, of Carnegie Mellon University, as well as our respective supervisors, Professor Michael Bowling (University of Alberta) and Professor J. Andrew Bagnell (Carnegie Mellon University). Much of the work presented in this thesis was originally described in conference proceedings that have the four of us as co-authors [41]. The original idea for two of the four new algorithms described in this thesis (regression regret-matching and regression counterfactual regret minimization) are due to Waugh. Theorems 3.0.3, 3.0.4, 3.0.7, and Corollary 3.0.5 were originally proven by Waugh. However, the proofs in this thesis fix a mistake in these original proofs. The proofs in this thesis are also greatly expanded, and Theorem 3.0.3 explicitly shows the proof of a previously omitted edge case. Some of the experimental results I contributed to the paper with Waugh et al. [41] are reproduced in Chapter 4 alongside new results and analysis.

It's one of the fundamental principles of programming, that it's extremely difficult to gauge how much work is hidden behind the statement of a task, even to where the trivial and impossible look the same when silhouetted in the morning haze.

– James Hague, "If You Haven't Done It Before, All Bets Are Off", *Programming in the Twenty-First Century* (http://prog21.dadgum.com/209.html).

Acknowledgements

I would like to thank

- my supervisor, Professor Michael Bowling, for his time, guidance, and advice.
- Professor Duane Szafron, who introduced me to algorithmic game theory and involved me in the CPRG during my undergraduate studies.
- the Computer Poker Research Group (CPRG) for software infrastructure support.
- the current members of the CPRG: Michael Johanson, Neil Burch, Nolan Bard, Kevin Waugh, Trevor Davis, Professor Rob Holte, and Viliam Lisy.
- past members of the CPRG: Josh Davidson, Richard Gibson, and Johnny Hawkin.
- fellow graduate students Marlos Machado, Zaheen Ahmad, and Tim Yee for research discussions.
- Professor Paul Lu, who was my first computing science teacher and who has since provided me with tremendous support and advice.
- Professor Jim Hoover, who initially convinced me that I could be successful in computing science when I began university.
- my fiance, Melanie Jamieson for her love, support, and companionship.
- my family, particularly my parents, Rosalie and Rick Morrill, for their constant encouragement and willingness to help.
- the Natural Sciences and Engineering Research Council of Canada (NSERC) and Alberta Innovates Technology Futures (AITF) for the Canada Graduate Scholarship (CGS) and Alberta Innovates Graduate Student Scholarship respectively, that helped to fund this thesis.
- Alberta Innovates Centre for Machine Learning (AICML) for providing the remainder of funding for this thesis.
- Calcul Québec, Westgrid, and Compute Canada for computing resources on which experiments were run.

Table of Contents

List of Figures

Chapter 1

Introduction

Game theoretic solution concepts, such as Nash equilibrium strategies that are optimal against worst case opponents, provide guidance in finding desirable autonomous agent behaviour. In particular, we wish to approximate solutions to complex, dynamic tasks, such as negotiation or bidding in auctions. Computational game theory investigates effective methods for computing such strategies. Solving human-scale games, however, is currently an intractable problem.

The typical approach when dealing with games too large to solve directly is to first create an **abstract game** that retains the same basic structure of the game one wishes to solve, but is a fraction of the size. Instead of solving the game directly, one solves this surrogate abstract game, and translates the abstract game solution back into the original game. The hope is that the abstract game inherited the original game's strategically important factors and the identified solution to the abstract game will perform well in the real game. While it has been extensively studied [33, 4, 17, 42, 21, 16, 31, 25, 23, 2, 26, 13], traditional abstraction is a flawed approach.

Waugh et al. [42] found that pathologies can exist where a solution to a finer abstraction is more exploitable against a worst-case opponent in the full game than a solution to a coarser abstraction. Imperfect information games in particular are difficult to abstract because parts of the game that are never reached and actions that are never used by an equilibrium may be necessary to prevent deviations from said equilibrium. The threat of deviating to an unused line of play might be necessary to punish the other player's deviation. These two properties make creating abstractions in imperfect information games problematic. As a result, crafting a good abstract

game requires care and extensive domain-specific knowledge.

But why are these games intractably large to begin with? Many real-world sequential decision problems such as online path planning [1], opponent exploitation [35], and portfolio optimization [18], as well as parlor games such as poker, typically have a regular structure and a compact description. Mapping these problems into a general form, such as the bandit setting or an extensive-form game, strips them of their structure and inflates their description, causing games like two-player, no-limit Texas hold'em poker to have more than $8.2 \cdot 10^{160}$ action choices over $2.7 \cdot 10^{160}$ decision points [22, p. 12][1].

In this thesis, we introduce an alternative to static abstraction by presenting a new online learning algorithm called **Regression Regret-matching (RRM)** that combines a dynamic, flexible abstraction, in the form of a **regressor**, with the elementary learning algorithm, **Regret-matching (RM)**. We show that RRM can be applied to compactly solve games when combined with **Counterfactual Regret Minimization (CFR)** without a preliminary abstraction step. For this new CFR variant, **Regression CFR (RCFR)**, we derive theoretical guarantees[2] akin to CFR's, except that RCFR's solution approximation bound depends on the accuracy of its regressor. The regressor takes advantage of the latent structure of the game through a **feature representation** that decomposes and factorizes game sequences. When there are strong relationships or redundancies between sequences, the regressor's representation of the game can be a fraction of the size of the game without severely degrading the solution quality.

In addition, we incorporate the recent work of Tammelin [37] to create RRM$^+$ and RCFR$^+$, which are variants of RRM and RCFR respectively. We prove that RRM$^+$ and RCFR$^+$ inherit the same theoretical properties as RRM and RCFR, and show that RRM$^+$'s **regression problem** allows the full representational power of its regressor to improve its strategy approximation. Because of this, RRM$^+$'s regression problem may be easier than RRM's, thereby allowing RCFR$^+$ to find less exploitable

[1] This is the size of the 50-100 blinds, 20,000 chip stacks as played in the Annual Computer Poker Competition (ACPC).

[2] Our derivation includes a correction to the originally published RCFR proof with Waugh et al. [41].

strategies with simpler representations.

We tested these RCFR variants in two small artificial poker games, comparing the exploitabilities of the strategy profiles they output against each other, as well as those found by running CFR on traditional abstract games. One of the games is a limit poker game where only one fixed bet size is allowed at any point in the game and the game complexity comes from the number of possible chance outcomes. The other is a no-limit game where its complexity derives from the many betting options available to each player. These experiments provide evidence that RCFR$^+$, and to a lesser extent RCFR, are indeed more effective than traditional abstractions.

Chapter 2

Background

Before describing RCFR and its variants, we first present the formal setting and terminology of extensive-form games, online learning, and supervised learning, as well as the rules of the poker games that we use as test domains.

Extensive-form Games

An **extensive-form game** is a model of games that includes sequential decisions and stochastic events. A play-out of a game is formed by walking a directed tree from its root to a leaf, where edges are actions by players or chance and nodes are game states. The leaf where a play-out ends contains that play-out's utility allocation for all players. The states where a player must act are partitioned into information sets such that all states in the same information set are indistinguishable to the acting player. All information sets are singletons in a perfect information game, such as chess, but some are non-singletons in imperfect information games, such as poker.

Formally,

Definition 2.0.1 (Osborne and Rubinstein [29])**.** *A **two-player, zero-sum, extensive-form game**, is a tuple $\Gamma = (\mathcal{H}, p, \sigma_c, \mathcal{I}, u)$ [29], where*

- *$\mathcal{H}$ is the set of game **histories**, which form a tree rooted at the empty history, $\emptyset \in \mathcal{H}$.*
- *$A(h)$ is the set of actions available at $h \in \mathcal{H}$, and $ha \in \mathcal{H}$ for each $a \in A(h)$ are children of h. In addition, let the set of **terminal histories**, $\mathcal{Z} \subseteq \mathcal{H}$ such that $z \in \mathcal{Z}$ if and only if $|A(z)| = 0$.*

- $p\colon \mathcal{H} \setminus \mathcal{Z} \to \{1, 2, c\}$ *is the* ***player choice function*** *that determines the next player, including a* ***chance player****, c, to act after any given non-terminal history.*

- $\sigma_c(a|h) \in \Delta_{A(h)}$[1] *for* $\{ h \in \mathcal{H}\colon p(h) = c \}$ *and* $a \in A(h)$*, is a fixed probability distribution over chance outcomes.*

- $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$ *is the* ***information partition****, which groups histories into* ***information sets*** *where all histories in an information set are indistinguishable to the acting player and have the same action sets.*

- $u\colon \mathcal{Z} \to \mathbb{R}$ *is the* ***utility function*** *that associates a value to each terminal history. Let* $u_1(z) = u(z)$ *be the utility of player 1 and* $u_2(z) = -u(z)$ *be the utility of player 2 at terminal history* z*.*

A **behavioral strategy for player i**, $\sigma_i \in \Sigma_i$, defines a probability distribution at all information sets where player i acts. That is, if $I \in \mathcal{I}_i$, then $\sigma_i(\cdot|I) \in \Delta_{A(I)}$. We call a tuple of strategies (σ_1, σ_2) a **strategy profile**. Let $\pi^\sigma(z)$ be the probability of reaching z by traversing the game tree with both players following σ from the root. Let $\pi^\sigma_{-i}(z)$ be the probability of reaching z using σ assuming player i takes actions to reach z with probability one. Let $\pi^\sigma(h, z)$ be the probability of reaching z using σ from history h, where $\pi^\sigma(h, z)$ is zero if h is not an ancestor of z. The **expected utility to player i** under profile σ can then be written as $u_i(\sigma) = \sum_{z \in \mathcal{Z}} \pi^\sigma(z) u_i(z)$.

Equilibria

An **ε-Nash equilibrium** is a strategy profile where neither strategy can unilaterally deviate to gain more than ε utility. That is,

$$\begin{aligned} u_1(\sigma_1, \sigma_2) + \varepsilon \geq u_1(\sigma_1', \sigma_2), \text{ and} \qquad & \forall \sigma_1' \in \Sigma_1 \\ u_2(\sigma_1, \sigma_2) + \varepsilon \geq u_2(\sigma_1, \sigma_2') \qquad & \forall \sigma_2' \in \Sigma_2. \end{aligned}$$

An **equilibrium strategy**, that is, a strategy part of a Nash equilibrium, is **minimax optimal** in a two-player, zero-sum game. This means that such strategies maximize

[1] $\Delta_S \subset \mathbb{R}^{|S|}$ denotes the probability simplex over set S, so a distribution $\sigma \in \Delta_S$ must satisfy the conditions of a probability distribution: $0 \leq \sigma(s) \leq 1$ for all $s \in S$ and $\sum_{s \in S} \sigma(s) = 1$.

their utility against a worst-case opponent and are low-risk strategies to play when nothing can be assumed about the population of players one would compete against.

Online Learning

The standard online learning setting is a general repeated game framework. A learner must choose a strategy, $\sigma^t \in \Delta_{|A|}$ (a probability distribution over actions, A), on every round and attempt to maximize her utility when the utility function is specified by a potentially omnipotent adversary. In each round $t \in [T] = [1, \ldots, T]$, the learner chooses a strategy, then observes the outcome of the game in the form of a bounded utility vector, v^t, where $\|v^t\|_\infty \leq L$. Each element v_a^t corresponds to the utility that the learner would have received for putting all of its mass on action $a \in A$. The learner receives $\sigma^t \cdot v^t = \sum_{a \in A} \sigma_a^t v_a^t$ as its utility on round t, then updates its strategy for the next round. The learner's final score is its cumulative utility, $\sum_{t=1}^{T} \sigma^t \cdot v^t$, so the question posed to algorithm designers is, "how should the learner adapt its strategy on each round to improve its score?"

Regret minimization is a well studied and effective answer. Regret is defined abstractly as the difference between the learner's cumulative utility and the cumulative utility she could have received if her strategies had been modified in a systematic way. Of particular interest is **external regret** where the modification is to always play the best action in hindsight over all rounds. If we think about the **instantaneous regret**, r_a^t, of action a on round t—that is, the difference between the utility of playing a with 100% probability and the realized utility on round t—then we can write external regret, $R^{\text{ext},T}$, as the maximum over a **cumulative regret** vector:

$$R^{\text{ext},T} = \max_{a \in A} R_a^T = \sum_{t=1}^{T} r_a^t = \sum_{t=1}^{T} v_a^t - \left(\sigma^t \cdot v^t\right) \mathbf{1}, \text{where } \mathbf{1} \text{ is the vector of all ones.}$$

We say that a learner is **no-regret** if her regret grows sublinearly with T ($R^{\text{ext},T} \in \mathrm{o}(T)$) so that her average regret approaches zero as T increases. Now imagine that two learners repeatedly compete in a zero-sum game for T rounds. It is well known that the round-by-round average of the strategies used by these learners form a 2ε-equilibrium, given that the average external regret of either player is no more than

ε. Thus, the average strategies of two no-regret learners converges to an equilibrium, and the accuracy with which an equilibrium is approximated can be chosen arbitrarily by selecting a sufficiently large T.

Regret-Matching

Regret-matching (RM) is an elementary example of a no-regret algorithm. First, notice that any mapping between actions and non-negative weights, $\omega \in \mathbb{R}^{|A|,+}$, admits the elementary strategy,

$$\sigma \coloneqq \begin{cases} \frac{\omega}{\|\omega\|_1}, & \text{if } \|\omega\|_1 = \sum_{a \in A} \omega_a > 0 \\ \frac{\mathbf{1}}{|A|}, & \text{otherwise.} \end{cases} \tag{2.1}$$

If one desires to use this method to generate a strategy but also wishes to use a mapping $w \in \mathbb{R}^{|A|}$ that could include negative weights, w can simply be projected onto the positive orthant: $\omega \coloneqq w^+ = \max\{\mathbf{0}, w\}$. RM is the application of Method 2.1 where the cumulative regret vector up to the previous round, $t-1$, is used as its weight mapping ($w \coloneqq R^{t-1}$, $R^0 \coloneqq \mathbf{0}$). A learner that uses RM has her external regret bounded by $L\sqrt{T\,|A|}$, thus RM is no-regret [5].

RM$^+$

Regret-matching$^+$ (RM$^+$) [37] is a simple modification of RM that yields a no-regret algorithm with useful properties. While RM projects potentially negative cumulative regrets onto the positive orthant to generate its strategy, RM$^+$ prevents its weights from becoming negative. RM$^+$ stores regret-like values, $Q^t \in \mathbb{R}^{|A|,+}$, where $Q^t = (Q^{t-1} + r^t)^+$ and $Q^0 \coloneqq \mathbf{0}$, hereby denoted **cumulative Q-regrets**, and applies Method 2.1 with $\omega \coloneqq Q^{t-1}$ to generate its strategy on round t. As a result, RM$^+$ will always play an action after it receives a positive instantaneous regret, no matter how poor its past returns.

Tammelin et al. [38] not only proved that RM$^+$ shares RM's $L\sqrt{T\,|A|}$ external regret bound and is therefore no-regret, but also that RM$^+$ achieves a sublinear **tracking regret**. Rather than considering the regret baseline to be a single action over all T rounds, tracking regret allows the best action to switch $(k-1)$ times, where k is a factor in the bound. Thus, RM$^+$ is no-regret even in non-stationary

settings where the utility distribution for each action may be changing over time. RM$^+$ is the first RM variant to be shown to have this property.

Note that since Q-regrets are guarenteed to be non-negative by definition, there is no need to project Q-regrets onto the positive orthant before generating a strategy. 0 is the only Q-regret value that can cause an action to receive no probability mass in the resulting strategy. Compare this to RM, where any non-positive regret will have the same effect. For example, in a two action game after T rounds, $R^T = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $R^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ will both result in the same strategy, $\sigma^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, showing that the exact value of a non-positive regret is inconsequential to the resulting strategy. We will return to this difference between RM and RM$^+$ in the context of **Regression Regret-matching (RRM)** to motivate a combination of RRM and RM$^+$ and explain its potential advantage over RRM.

Counterfactual Regret Minimization

Potentially, one could use any applicable no-regret algorithm, such as RM, to solve any two-player, zero-sum game, except that this would be intractable for games with many actions. Extensive-form games in particular present a challenge for these algorithms as they would need to compute and store the regrets for each deterministic strategy, of which there are exponentially many in the number of information sets.

To avoid these problems, Zinkevich et al. [43] developed **Counterfactual Regret Minimization (CFR)**, which requires memory and computation only proportional to the number of information sets.

For each player $i \in \{ 1, 2 \}$, define

- the **counterfactual value** for taking action $a \in A(I)$ in information set $I \in \mathcal{I}_i$ to be the expected utility of a given that I is reached, weighted by the probability that all other players and chance play to reach I,

$$v_i^t(a|I) = \sum_{h \in I} \sum_{z \in \mathcal{Z}} \pi_{-i}^t(h) \pi^t(ha, z) u_i(z).$$

- the **instantaneous counterfactual regret** of action $a \in A(I)$ in information set $I \in \mathcal{I}_i$ on round t to be the external regret with respect to counterfactual

values,

$$r_i^t(a|I) = v_i^t(a|I) - \sum_{a' \in A(I)} \sigma_i^t(a'|I) v_i^t(a'|I).$$

- the **cumulative counterfactual regret** of action $a \in A(I)$ in information set $I \in \mathcal{I}_i$ on round T to be the sum of instantaneous counterfactual regrets on the same sequence over T rounds,

$$R_i^T(a|I) = \sum_{t=1}^{T} r_i^t(a|I).$$

CFR relies on the sum of player i's positive counterfactual regrets over each information set being an upper bound on the player's external regret, $R_i^{\text{ext},T}$. Formally, $R_i^{\text{ext},T} \leq \sum_{I \in \mathcal{I}_i} R_i^{T,+}(I)$, where $R_i^T(I) = \max_{a \in A(I)} R_i^T(a|I)$, and this was proven by Zinkevich et al. [43]. It follows immediately that minimizing the counterfactual regret at each information set independently will also minimize a player's external regret, thereby ensuring that the average over time of the CFR players' strategies converges to a Nash equilibrium.

CFR places independent no-regret learners at each information set, and each works to minimize the counterfactual regret at its associated information set. On every iteration, the learners together produce a behavioral strategy profile, called the **current strategy profile**, from which counterfactual values are computed. Counterfactual values induce counterfactual regrets, which are used to update each of the learners. The realization plan of the current strategies are averaged over each iteration to form the **average strategy profile**. Since the counterfactual regret, and thus the external regret, of both players in the CFR algorithm is approaching zero, the average strategy profile is an increasingly accurate approximation of an equilibrium as T increases.

Strategies generated by variants of CFR currently dominate the **Annual Computer Poker Competition (ACPC)**, both in terms of the number of entrants and their performance.

The original description of CFR updated the regrets for both players simultaneously [43], but more recently it has been found that alternating these updates makes

CFR variants perform better in practice [38], so all CFR variants implemented for this thesis alternate their updates.

CFR$^+$

Swapping out RM for RM$^+$ in CFR results in CFR$^+$, which was recently used to solve heads-up limit Texas hold'em [6]. While current theory shows that its worst-case performance is the same as CFR's [38], CFR$^+$ converges substantially faster in practice [6].

The linearly weighted averaging $\bar{\sigma}_p^T = 2/(T^2 + T) \sum_{t=1}^{T} t\sigma_p^t$, as used by Tammelin et al. [38], allows the average strategies to adapt faster, thereby allowing them to be less exploitable after fewer iterations if the current strategies also tend to have low exploitabilities. Tammelin et al. showed that CFR$^+$ with linear averaging still converges to an equilibrium, while the same property was not shown for plain CFR. Linear averaging can also hurt CFR's performance in practice [38], so there is no reason to use it instead of uniform averaging in CFR. Since empirical convergence rates are not the subject of this thesis, the CFR and CFR$^+$ variants implemented for this thesis used uniform averaging for simplicity and consistency.

Poker Games

The field of artificial intelligence has a rich tradition of using games as proving grounds for its methods, including poker, chess, and go. Poker stands out among these testbeds as a setting that includes both imperfect information and chance. The notions of bluffing and misdirection, which are synonymous with poker, were integral to the development of game theory, according to a conversation between Bronowski and von Neumann [6]. In addition, poker can be scaled by altering its rules, such as changing the composition of the deck, the number of betting rounds, or the number of chips in each player's stack. Scaling different portions of poker games can also allow experiments that better emphasize different aspects of a learning system, such as the way it deals with a wider range of chance events or available betting actions. We will use two poker variants in our empirical validation.

Leduc Hold'em

Leduc hold'em [34] is a poker game based on Kuhn poker. It is a convenient testbed as common operations, like best response and equilibrium computations, are tractable and exact.

The game has two betting rounds, the preflop and flop. At the beginning of the game both players ante a single chip into the pot and are dealt a single private card from a shuffled deck of six cards—two jacks, two queens and two kings. Then begins the preflop betting round where the first player can either check or bet. If the first player checks, passing their turn, then the second player can end the betting round by checking as well, or continue by betting. When facing a bet; the player can raise by placing two chips into the pot; call by matching the bet in the pot and ending the round; or fold by forfeiting the pot to the opponent. There is a maximum of two wagers per round, *i.e.*, one bet and one raise. A single public card is dealt face up for both players to see at the beginning of the flop. If the flop betting ends without either player folding, a showdown occurs and the player with the best hand takes the pot. A player that pairs, *i.e.*, their card matches the public card, always has the best hand no matter the rank of the paired card or the opponent's card. If neither player pairs, the one with the highest rank card wins. In the event of a tie, the pot is split. The size of a wager preflop is two chips, and is doubled to four chips on the flop.

Leduc hold'em has 672 canonical sequences[2]. At equilibrium, the first player is expected to lose 0.086 chips per hand. The typical utility measure in poker games is the millibig-blind (mbb), which is a thousandth of the big-blind. In Leduc hold'em, both players pay an ante of one chip, so the big-blind is effectively one chip, and the first player loses 86 mbb optimally.

No-limit One-card Poker

Most of the size of Leduc hold'em comes from the size of its deck, the presence of a public board card, and the number of rounds. But other interesting games result from modifying Kuhn poker by increasing the number of betting actions available by

[2]The sequences that cannot be merged by identifying card suit symmetries.

increasing each player's chip stack size, and making the game **no-limit**. A no-limit poker game is one where players are allowed to wager any number of chips between a minimum wager and the acting player's stack size, but is otherwise the same as limit poker.

The second game we use in this thesis is a variant of no-limit one-card poker with one betting round, a four card deck where each have the same suit but different rank, no public cards, ten chip stacks, and a one chip ante for both players. Thus, all matches begin with a pot containing two chips and both players have nine chips to spend to wager back and forth until a player calls or folds to end the match. While at first glance this game may appear simpler than Leduc hold'em since it has fewer rounds and a smaller, simpler deck, this one-card poker game has 3200 sequences, making it more than four times larger.

Similar to Leduc hold'em, the big-blind is effectively its one chip ante, and the first player is expected to lose 83 mbb optimally in this game.

Scaling CFR

CFR is efficient in the number of information sets, but when the game has more than 10^{160} of them, like no-limit Texas hold'em, CFR is impractical. Fortunately, CFR lends itself readily to various sampling and abstraction techniques that improve its computational properties, perhaps at a cost to the solution approximation quality. As we are proposing an alternative approach to scaling CFR, we briefly review past approaches to this problem.

Sampling

A common technique for reducing the computation of statistical estimations is sampling. It is the fundamental idea used by Monte-Carlo tree search algorithms, which are state-of-the-art players of perfect information games like go [10, 14] and hex [19]. Monte-Carlo CFR (MCCFR) [28], an extension of CFR, was also developed to use sampling to solve large games more quickly. MCCFR represents a family of CFR variants that sample player or chance actions to obtain estimates

of counterfactual values, and thus regrets. These sampled counterfactual regrets are unbiased estimates of their unsampled counterparts, so the CFR convergence theory holds in expectation. Average strategy sampling [8], external sampling [28] and public chance sampling [24] typically outperform their sister variants, including vanilla CFR, in terms of running time in practice. Average strategy sampling is the least used of these three however, as it requires the most parameter tuning.

Abstraction

A popular technique to generate approximate equilibrium strategies in games too large to solve directly, even when sampling, is to solve a smaller abstracted version of the game and play the solution in the original game [33]. Games are typically abstracted in two ways: merging information sets in an information or **state-space abstraction** [4, 17, 21, 26] and doing an **action abstraction** by discretizing or shrinking the set of legal actions [16, 31, 13].

State-space Abstraction

Two or more information sets, $I_1, I_2, \ldots, I_n \in \mathcal{I}$, that are compatible, in the sense that their action sets are the same ($A(I_1) = A(I_2) \ldots = A(I_n)$), can be collapsed into a single information set in an abstract game. The information sets involved in the merge were distinct in the original game, but are indistinguishable in the abstract game. More formally, one may define this type of abstraction as a many-to-one mapping, $f : I^{\text{full}} \rightarrow I^{\text{abstract}}$, where I^{full} is the set of information sets in the full game, and I^{abstract} is that in the abstract game.

In this way, an abstract game is built to be a smaller version of the original game. For example, in Leduc hold'em, merging all the information sets where the player holds either the king or queen allows the player to distinguish between holding a jack versus a queen, or a jack versus a king, but not distinguish between holding a queen versus a king. The hope is that there is redundancy in (near) equilibrium strategies of the original game that are leveraged by the abstraction: strategies with low exploitability that, for all $I \in I^{\text{abstract}}$, use a similar policy at all $I' \in I^{\text{full}}$ where $f(I') = I$ [27]. Of course, to find such strategies they must be equilibrium strategies

in the abstract game, so the design of the abstraction would ideally reflect this. A state-space abstraction is often generated by specifying a similarity function between compatible information sets and grouping them with a clustering algorithm, such as k-means [26], where the size of the resulting abstraction is the number of clusters.

Action Abstraction

Instead of grouping previously disjoint information sets, one could remove information sets by restricting the action sets of each player. Considering now a no-limit poker game, allowing players to take only fold, call, pot size wagers, and all-in wagers would be an example of such an abstraction.

A benefit of removing actions is that it prevents agents from playing risky actions that would require a finely-grained abstraction to properly utilize or exploit. For example, a small bet in no-limit poker increases the risk and potential utility of the hand and prolongs the betting round. To be effective, the player making the small bet must be able to make a good decision if the opponent re-raises. Many good decisions have to be made in a row for the small bet to pay off, and otherwise the player takes a greater loss than she would have if she would not have bet. In contrast, an all-in wager is always the player's final action, and call or fold are the only options available to the opponent for her final action. An all-in wager has a positive expected value as long as the actor's hand defeats more than half of the hands that the opponent could be holding, averaged over all potential board cards. So deciding whether to make an all-in versus a check/call or fold is simple, it limits the opponent's options, and it requires no additional reasoning to obtain its payoff, thus making all-in a low risk option.

An additional complication involved in using an action abstraction is that a translation step is required before the solution to the abstract game can be played in the original game. Because play-outs in the original game may involve actions that were declared illegal in the abstract game, the real information set must be translated into its closest counterpart in the abstract game [31]. The abstract game solution can then sample its behavioral strategy at this abstract information set to select an action.

Supervised Learning

The modifications to RM and CFR presented in this thesis center around the idea that a table of regrets or similar values can be faithfully reproduced by a more compact representation. To describe these modifications and explain how one would go about building such a representation, we first give an overview of **supervised learning**.

Supervised learning refers to a particular learning setting where the learner must produce a function that generalizes a batch of labeled examples. The learning is "supervised" because training examples are labeled, *i.e.* every input example has an associated output value, also called a **target**. In contrast to online learning, we typically think about supervised learning as being done in a one-shot manner where the learner observes all input–target pairs before producing any output. The output of a supervised learning algorithm is a mapping from instances of the input space to instances of the target space. The goal is to produce a mapping that compactly represents the underlying relationship between training inputs and targets. With such a mapping, one could reproduce training targets from their inputs and accurately estimate the values of previously unseen inputs.

If the targets are continuous scalars, the problem is called a **regression problem**. For example, finding the least squares fit between two-dimensional data points, where one dimension is the input space and the other is the target space. An algorithm that solves a regression problem is called a **regressor**. A regressor produces a function, $f\colon \mathcal{X} \to \mathbb{R}$, that maps elements of the **input space**, $\mathcal{X}$, to a real number.

When $\mathcal{X}$ is complicated, *e.g.* the space of images or the sequences of a game, we can define a **feature space** and an associated **feature expansion** for all $x \in \mathcal{X}$, $\varphi(x) \in \Phi$, that describes characteristic properties of elements in $\mathcal{X}$. Typically, the feature space is Euclidean, *i.e.* $\Phi \subseteq \mathbb{R}^m$, where $m > 0$ is the number of features, and $\varphi(x)$ is called the **feature vector** of the instance $x \in \mathcal{X}$. Features that could be used to describe the sequences of a poker game could be the size of the pot, the size of the wager to be made, the round, the next player to act, whether or not the acting player is holding a king, or whether the acting player's private card pairs a community board card. Once a feature expansion is defined, any of a number of

supervised learning algorithms could be applied to learn a mapping between features and targets.

Regression Tree

One such prediction model is a regression tree. A regression tree is a piecewise-constant function that is induced by a hierarchy of questions, or **splits**, that determine the estimate for any $x \in \mathcal{X}$. Given $\varphi(x)$, the tree begins by checking the feature values against the root node's question, then takes the path determined by the answer. Typically, regression trees are binary, so the left path will be taken if the answer is affirmative or the right if it is negative. If the next node encountered is a leaf, that leaf's value is returned as the prediction. If it is an internal node, the features are checked against this node's splitting feature, and the appropriate path is again taken.

Each leaf value is a function of the targets associated with the training examples that would fall into the leaf. For a regression tree that attempts to minimize the squared error from the training data, the mean of the targets in the leaf is used as the leaf value. Alternatively, the median is chosen to minimize absolute error. The questions could be chosen in different ways, but two common ways are by recursive [7] or best-first partitioning [12, 32].

Recursive Partitioning

To build a regression tree with recursive partitioning, the examples in each node are split on the feature value that would most reduce the average error in that node. At the current node on the frontier of the growing regression tree, beginning at the root, the split is chosen that induces the partition that minimizes the average error of the data in the node. Two child nodes are created, one for each side of the partition, and this process is repeated in each of the children. Splits are made until there are no more splits to make or a stopping condition has been reached. Two common stopping criteria are that the error reduction of the next split in proportion to the unsplit error is below a threshold, or the average error within the node is less than a threshold[3].

[3]Waugh et al. [41] states that a threshold on the proportional error reduction was used as the stopping condition, but this is inaccurate. It was in fact a threshold on the average error within the node.

The value of such thresholds would then determine the granularity of the partition induced by the regression tree and the tree's size.

Best-first Partitioning

A regression tree is built with best-first partitioning by progressively making the best split considering all the nodes on the frontier of the tree. The best split is the one that most reduces the error of the tree overall. The reduction in error that would result from making a split is equal to the difference between the error between the examples in the node and the sum of the errors of the potential child nodes, so this can be computed efficiently. Whenever a split is made, the best splits of the new child nodes are computed and saved to be compared with the error reductions of the previously made splits on the frontier of the current tree. Deciding which split to make only requires searching through the previously computed error reductions, one for each frontier node, to find the best split.

Splits would continue to be made until making a split becomes too expensive according to a **regularization function**. For example, a regularizer that returns the current size of the tree increases the cost of splitting a node linearly as the tree grows, and would encourage the creation of smaller trees. Analogous to the regularizer of a linear model, such biases toward simple tree models could be useful to improve generalization and avoid overfitting.

In the context of a supervised learning problem where the learner has a parameter or complexity budget, recursive partitioning is inconvenient because the error threshold must be tuned to meet the budget. Finding an acceptable threshold automatically requires building multiple trees with various thresholds in a parameter search, which is computationally wasteful. Meeting a complexity budget with best-first partitioning on the other hand, is trivial, since one can set the regularizer to be zero until the budget is exhausted, and infinite afterwards.

Motivation for Using Regression Trees

While it is NP-complete to find an optimal regression tree [20], the non-linearity of the partitions induced by the tree structure facilitates the learning of complicated

functions. The tree structures learned in this fashion are a rudimentary but often effective form of representation learning. While slower to train than a linear regressor, trees are fast to train and evaluate compared to more complicated models, such as kernel representations or deep networks. The dominating computational task is sorting the training examples in a node to be split, which typically takes $\mathcal{O}(n \log n)$ operations, where n is the number of examples to sort, and this must be done once for every feature [39]. Evaluating a new example costs $\mathcal{O}(s)$ in the worst-case, where s is the number of splits in the tree, but the estimate could potentially return after checking the first split.

Chapter 3

Functional Regret Estimation

Here we describe new algorithmic and theoretical contributions to the fields of online learning and game solving. The key idea that leads to the following algorithms is that we can view traditional abstraction as one choice of a compact representation for an intractably large tabular mapping between game sequences and values (regrets or Q-regrets, typically). A regressor is a different way of compactly representing an arbitrary mapping, one that is more flexible and adapts to training examples. So instead of thinking about abstracting a game and subsequently solving the abstract game with a technique like CFR, we can think about defining features on the original game sequences and estimating associated values. Such an estimator can be trained on values that are computed during each CFR iteration. This intuition forms the basis of **Regression RM (RRM)** and **Regression CFR (RCFR)**.

Regression RM

Regression RM (RRM) is an online learning algorithm that is a principled generalization of RM. RRM allows structural information between actions to be used to improve learning and compactly represent regrets. Structural information here is in the form of a feature expansion, $\varphi(a) \in \Phi$, that is associated with every action $a \in A$. Intuitively, if two actions $a \in A$ and $a' \in A$ have similar features, then taking action a and receiving a utility should provide information about what the utility would be if a' were taken, and vice-versa. Note that one can reduce RRM to RM by using indicator features to describe each action and a regressor that averages the

targets associated with each action, such as gradient descent with a $\frac{1}{t}$-learning rate.

After every round of an online learning problem, the learner receives a utility vector v^t, which admits an instantaneous regret vector, $r^t = v^t - (\sigma^t \cdot v^t)\mathbf{1}$. Typically, RM would accumulate these instantaneous regrets in a cumulative regret vector, $R^t = \sum_{s=1}^{T} r^s$, and use R^t to make an informed prediction on subsequent rounds. Instead, RRM uses a regressor to learn a function, $f^t \colon \Phi \to \mathbb{R}$, where $f^t(\varphi(a))$ approximates the average instantaneous regret, $\bar{r}_a^t = \frac{1}{t}\sum_{s=1}^{t} r^s$. An estimate of the cumulative regret is then $R_a^t = t\bar{r}_a^t \approx \tilde{R}_a^t = tf^t(\varphi(a))$. RRM's policy on round t, σ^t, is then proportional to the regret estimates after the previous round, $\tilde{R}^{t-1}$.

In a stateful setting, where many learners are used to find good strategies in each state independently, RRM allows features of the state, or state–action combinations, to be specified so that a regressor can estimate the regrets for each state–action.

RRM has an advantage over the conventional approach when there is structure between states and actions. For example, in a poker game, holding any two different sets of cards represents two distinct information sets, but they might in fact be strategically similar, or two wager sizes may be close enough that they could be considered the same. More generally, if an elementary relationship exists between states and actions, RRM may allow the collection of strategies to improve faster and with lower storage requirements. And rather than defining a static abstraction before learning, RRM allows a dynamic abstraction, in the form of a regressor, to be shaped by the data obtained while learning.

Regret Bounds

The regret of RRM can be bounded in terms of the representational power of the regressor. The proof of which requires the following general definitions and theorems.

Definition 3.0.2 (ϵ-Blackwell's Condition)**.** *Let*

$$\phi(b) = (b^+)^2 \textit{ for } b \in \mathbb{R}\textit{, and}$$
$$\Phi(y) = \sum_{i=1}^{d} \phi(y_i) = \left\|y^+\right\|_2^2 \textit{ for } y \in \mathbb{R}^d \textit{ and } d > 0.$$

Then Blackwell's condition is satisfied with $\epsilon \in \mathbb{R}$ error if

$$\sup_{v^{t+1} \in \mathcal{V}} \nabla\Phi(R^t) \cdot r^{t+1} \leq \epsilon. \tag{3.1}$$

Theorem 3.0.3 (Bound on Error in Achieving Blackwell's Condition)**.** *Consider the online learning setting as described in Section 2 with action set A. A learner that uses Method 2.1 with weights $\omega \coloneqq y^+$, where $y \in \mathbb{R}^{|A|}$, to generate her strategy on round $t+1$, satisfies ϵ-Blackwell's condition with ϵ at most*

$$4 \left\|v\right\|_2 \left\|R^{t,+} - y^+\right\|_1{}^1. \tag{3.2}$$

Proof. Consider arbitrary $v = v^{t+1} \in \mathcal{V}$. Expanding and doing algebra on the left-hand-side of Blackwell's condition,

$$\begin{aligned}
\nabla\Phi(R^t) \cdot r^{t+1} &= \nabla\Phi(R^t) \cdot [v - (v \cdot \sigma)\,\mathbf{1}] && (3.3)\\
&= 2\left[R^{t,+} \cdot v - R^{t,+} \cdot [(v \cdot \sigma)\,\mathbf{1}]\right] && (3.4)\\
&= 2\left[R^{t,+} \cdot v - \left(R^{t,+} \cdot \mathbf{1}\right)(\sigma \cdot v)\right] && (3.5)\\
&= 2\left[R^{t,+} \cdot v - \left(\left\|R^{t,+}\right\|_1 \sigma\right) \cdot v\right] && (3.6)\\
&= 2v \cdot \left(R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right). && (3.7)
\end{aligned}$$

Note that (3.7) is always zero when $R^{t,+} = \mathbf{0}$, so the theorem is trivially proven in this case. Consider now only the case when $R^{t,+} \neq \mathbf{0}$. By Cauchy-Schwarz,

$$2v \cdot \left(R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right) \leq 2\left\|v\right\|_2 \left\|R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right\|_2 \tag{3.8}$$

Since $\left\|x\right\|_2 \leq \left\|x\right\|_1$ for arbitrary $x \in \mathbb{R}^d$ and $d > 0$,

$$\leq 2\left\|v\right\|_2 \left\|R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right\|_1. \tag{3.9}$$

When $y = \mathbf{0}$, we can apply the triangle inequality to simplify the third term in the

product of (3.9),

$$\left\|R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right\|_1 \leq \left\|R^{t,+}\right\|_1 + \left\|\left\|R^{t,+}\right\|_1 \sigma\right\|_1 \tag{3.10}$$

$$= \left\|R^{t,+}\right\|_1 + \left\|R^{t,+}\right\|_1 \|\sigma\|_1 \tag{3.11}$$

$$= \left\|R^{t,+}\right\|_1 (1 + \|\sigma\|_1) \tag{3.12}$$

$$= \left\|R^{t,+}\right\|_1 \left(1 + \left\|\frac{\mathbf{1}}{|A|}\right\|_1\right) \tag{3.13}$$

$$= \left\|R^{t,+}\right\|_1 \left(1 + \frac{\|\mathbf{1}\|_1}{|A|}\right) \tag{3.14}$$

$$= \left\|R^{t,+}\right\|_1 (1 + 1) \tag{3.15}$$

$$= 2\left\|R^{t,+}\right\|_1. \tag{3.16}$$

Substituting (3.16) back into (3.9) finishes the proof for this case:

$$2\|v\|_2 \left\|R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right\|_1 \leq 2\|v\|_2 \left(2\left\|R^{t,+}\right\|_1\right) \tag{3.17}$$

$$= 4\|v\|_2 \left\|R^{t,+}\right\|_1. \tag{3.18}$$

When $y \neq \mathbf{0}$, we can apply the triangle inequality for distances to simplify the third term in the product of (3.9),

$$\left\|R^{t,+} - \left\|R^{t,+}\right\|_1 \sigma\right\|_1 = \left\|R^{t,+} - \left\|R^{t,+}\right\|_1 \frac{y^+}{\|y^+\|_1}\right\|_1 \tag{3.19}$$

$$= \left\|R^{t,+} - \frac{\|R^{t,+}\|_1}{\|y^+\|_1} y^+\right\|_1 \tag{3.20}$$

$$\leq \left\|R^{t,+} - y^+\right\|_1 + \left\|\frac{\|R^{t,+}\|_1}{\|y^+\|_1} y^+ - y^+\right\|_1 \tag{3.21}$$

$$= \left\|R^{t,+} - y^+\right\|_1 + \left\|\left(\frac{\|R^{t,+}\|_1}{\|y^+\|_1} - 1\right) y^+\right\|_1 \tag{3.22}$$

$$= \left\|R^{t,+} - y^+\right\|_1 + \left|\frac{\|R^{t,+}\|_1}{\|y^+\|_1} - 1\right| \left\|y^+\right\|_1 \tag{3.23}$$

$$= \left\|R^{t,+} - y^+\right\|_1 + \left|\frac{\|R^{t,+}\|_1}{\|y^+\|_1} \left\|y^+\right\|_1 - \left\|y^+\right\|_1\right| \tag{3.24}$$

$$= \left\|R^{t,+} - y^+\right\|_1 + \left|\left\|R^{t,+}\right\|_1 - \left\|y^+\right\|_1\right|. \tag{3.25}$$

By the reverse triangle inequality,

$$\leq \left\|R^{t,+} - y^+\right\|_1 + \left\|R^{t,+} - y^+\right\|_1 \tag{3.26}$$

$$= 2\left\|R^{t,+} - y^+\right\|_1. \tag{3.27}$$

Inserting (3.27) back into (3.9) completes the proof for this final case:

$$\nabla\Phi(R^t)\cdot(v-(v\cdot\sigma)\,\mathbf{1}) \le 2\left\|v\right\|_2\left(2\left\|R^{t,+}-y^+\right\|_1\right) \tag{3.28}$$

$$= 4\left\|v\right\|_2\left\|R^{t,+}-y^+\right\|_1. \tag{3.29}$$

□

Theorem 3.0.4 (Regret Bound Given ϵ^t-Blackwell's Condition is Satisfied)**.** *If a learner satisfies ϵ^t-Blackwell's condition on every round t, then the learner's regret is bounded by $\sqrt{T\left|A\right|L^2+\sum_{t=1}^T\epsilon^t}$. That is, so long as the learner chooses her σ^t such that $\sum_{t=1}^T\epsilon^t<T^2$, the learner will be no-regret.*

Proof. Adapted from Cesa-Bianchi and Lugosi [9]. Observe that $\nabla\Phi$ is 2-Lipschitz continuous:

$$\left\|\nabla\Phi(x)-\nabla\Phi(x')\right\|_2 = \left\|2x-2x'\right\|_2, \text{ for arbitrary } x,x'\in\mathbb{R}^d \text{ and } d>0 \tag{3.30}$$

$$= 2\left\|x-x'\right\|_2. \tag{3.31}$$

By Lipschitz continuity of $\nabla\Phi$,

$$\Phi(R^{t+1}) \le \Phi(R^t)+\nabla\Phi(R^t)\cdot\left(R^{t+1}-R^t\right)+\left\|R^{t+1}-R^t\right\|_2^2 \tag{3.32}$$

$$= \Phi(R^t)+\nabla\Phi(R^t)\cdot r^{t+1}+\left\|r^{t+1}\right\|_2^2. \tag{3.33}$$

By ϵ^{t+1}-Blackwell's condition,

$$\le \Phi(R^t)+\epsilon^{t+1}+\left\|r^{t+1}\right\|_2^2 \tag{3.34}$$

$$\le \Phi(R^t)+\epsilon^{t+1}+\sup_{v\in\mathcal{V}}\left\|v\right\|_2^2. \tag{3.35}$$

Since $\left\|x\right\|_2\le\sqrt{d}\left\|x\right\|_\infty$ for arbitrary $x\in\mathbb{R}^d$ and $d>0$,

$$\le \Phi(R^t)+\epsilon^{t+1}+\left|A\right|\left(\sup_{v\in\mathcal{V}}\left\|v\right\|_\infty\right)^2. \tag{3.36}$$

Since $\left\|v^t\right\|_\infty\le L$,

$$\le \Phi(R^t)+\epsilon^{t+1}+\left|A\right|L^2. \tag{3.37}$$

Iterating the inequality,

$$\le (t+1)\left|A\right|L^2+\sum_{s=1}^{t+1}\epsilon^s. \tag{3.38}$$

Bounding the maximum regret in terms of $\Phi(R^T)$, we find

$$\max_{a \in A} R_a^T \leq \max_{a \in A} R_a^{T,+} = \left\| R^{T,+} \right\|_\infty \tag{3.39}$$

$$\leq \left\| R^{T,+} \right\|_2 = \sqrt{\Phi(R^T)} \tag{3.40}$$

$$\leq \sqrt{L^2 T \left| A \right| + \sum_{t=1}^{T} \epsilon^t}. \tag{3.41}$$

□

Combining Theorems 3.0.3 and 3.0.4 allows us to prove a general regret bound that applies to RRM.

Corollary 3.0.5 (Regret Bound Given Regret Estimation Error in Terms of ℓ_1 Distance)**.** *A learner that uses Method 2.1 with weights $\omega \coloneqq y^{t,+}$, where $y^t \in \mathbb{R}^{|A|}$, to generate her strategy on round $t+1$, has her regret bounded by*

$$\sqrt{L\sqrt{|A|}\left(LT\sqrt{|A|} + 4\sum_{t=1}^{T} \left\| R^{t,+} - y^{t,+} \right\|_1\right)}. \tag{3.42}$$

So long as the learner chooses y^t such that $\sum_{t=1}^{T} \left\| R^{t,+} - y^{t,+} \right\|_1 < T^2$, the learner will be no-regret.

Proof. Setting $\epsilon^t \coloneqq 4 \left\| v \right\|_2 \left\| R^{t,+} - y^{t,+} \right\|_1$, as specified by Theorem 3.0.3, into 3.0.4 yields the desired inequality:

$$\sqrt{T \left| A \right| L^2 + \sum_{t=1}^{T} \epsilon^t} = \sqrt{L^2 T \left| A \right| + \sum_{t=1}^{T} 4 \left\| v \right\|_2 \left\| R^{t,+} - y^{t,+} \right\|_1} \tag{3.43}$$

$$= \sqrt{L^2 T \left| A \right| + 4 \left\| v \right\|_2 \sum_{t=1}^{T} \left\| R^{t,+} - y^{t,+} \right\|_1} \tag{3.44}$$

$$\leq \sqrt{L^2 T \left| A \right| + 4L\sqrt{|A|} \sum_{t=1}^{T} \left\| R^{t,+} - y^{t,+} \right\|_1} \tag{3.45}$$

$$= \sqrt{L\sqrt{|A|}\left(LT\sqrt{|A|} + 4\sum_{t=1}^{T} \left\| R^{t,+} - y^{t,+} \right\|_1\right)}. \tag{3.46}$$

□

Corollary 3.0.5 reduces to the familiar $L\sqrt{T \left| A \right|}$ bound when applied to RM. If one measures the deviation from regrets with the Euclidean distance, an extra $\sqrt{|A|}$ factor is added to the bound's error term, but the no-regret guarantee is unaffected.

Corollary 3.0.6 (Regret Bound Given Regret Estimation Error in Terms of Euclidean Distance)**.** *If the error between the cumulative regrets and the weights with which the learner is using to generate its strategy is measured as the Euclidean distance between these two vectors, the learner's regret is bounded by*

$$\sqrt{L\left|A\right|\left(LT+4\sum_{t=1}^{T}\left\|R^{t,+}-y^{t,+}\right\|_{2}\right)}. \tag{3.47}$$

Proof. Applying the fact that $\left\|x\right\|_1 \leq \sqrt{d}\left\|x\right\|_2$ for $x \in \mathbb{R}^d$ and $d > 0$ to Corollary 3.0.5 yields the desired bound. □

Algorithm Details

RRM has been defined abstractly, but the regression problem that must be solved to implement RRM has yet to be completely specified. In particular, what are the training targets? One choice is to make the training targets the instantaneous regrets, but on every round, the regressor either needs to be trained with all examples that have been observed since the first round, or it must adapt online to the latest regrets. Either way, this choice constrains the regressor to be one that predicts the average target given a specific feature vector, *i.e.* one that minimizes squared training error.

Algorithm 1 is an alternative that allows a regressor that does not minimize squared error. It requires storing and updating $\bar{r}^t$ to use as targets[2], which removes one of RRM's motivations and its potential advantages, but this might still be a worthwhile approach to gain sample-efficiency through generalization between actions, or simply to test that RRM would be useful in a particular problem with a particular regressor.

Algorithm 2 is a more practical approach that also allows one to use an arbitrary regressor. Unfortunately, it potentially introduces a compounding error problem, since it bootstraps the targets on round t with the predictions made by the regression function, f^{t-1} after round $t-1$. Since $(t-1)f^{t-1}(a) = \tilde{R}_a^{t-1} \approx R_a^{t-1}$, we can approximate $R_a^t \approx \tilde{R}_a^{t-1} + r_a^t$. Then RRM can train its regressor with $\varphi\left(a\right)$–$\frac{\tilde{R}_a^{t-1}+r_a^t}{t}$

[2] R^t could be used instead, but we discuss the version with $\bar{r}^t$ so that RRM's regressor is always estimating $\bar{r}^t$.

pairs, thereby bootstrapping itself, in the same sense as reinforcement learning's bootstrapping [36], *i.e.* its future predictions are based on its current estimates.

Since $\tilde{R}^{t-1}$ is an estimate that may not exactly match R^{t-1}, the bootstrapped target $\frac{\tilde{R}_a^{t-1}+r_a^t}{t}$ may not match $\bar{r}^t$, so errors incurred during the regression process will be felt twice on each round: once during the generation of $\tilde{R}^t$ to form the learner's policy, σ^t, and once during the creation of the bootstrapped targets. The error between R^t and $\tilde{R}^t$ depends not only on the regressor's capacity to reproduce its training data, but also the error between R^{t-1} and $\tilde{R}^{t-1}$, thereby causing small errors to possibly compound and become significant on later rounds. Our experiments in Chapter 4 investigate whether or not this is an issue in practice.

Algorithm 1 Regression Regret-Matching with Stored $\bar{r}$

$\bar{r}^0 \leftarrow \mathbf{0} \in \mathbb{R}^{|A|}$
$f \leftarrow$ CONSTANTFUNCTION(0) ▷ Always returns zero
for $t \in [T]$ **do**
  $\tilde{R}^t \leftarrow [tf^t(\varphi(a))]_{a \in A}$
  Choose $\sigma^t \propto \left(\tilde{R}^{t-1}\right)^+$
  Observe $v^t \in \mathbb{R}^{|A|}$
  $X^t \leftarrow []$
  for $a \in A$ **do**
    $X^t \leftarrow X^t \cup \varphi(a)$
    $r^t_{|A|} \leftarrow v^t_{|A|} - \sigma^t \cdot v^t$
    $\bar{r}^t_{|A|} \leftarrow \bar{r}^{t-1}_{|A|} + \frac{r^t_{|A|} - \bar{r}^{t-1}_{|A|}}{t}$
  end for
  $f^t \leftarrow$ TRAINREGRESSOR$(X^t, \bar{r}^t)$
end for

Algorithm 2 Regression Regret-Matching with Bootstrapping

$f \leftarrow \textsc{ConstantFunction}(0)$ ▷ Always returns zero
for $t \in [T]$ **do**
  $\tilde{R}^t \leftarrow [tf^t(\varphi(a))]_{a \in A}$
  Choose $\sigma^t \propto \left(\tilde{R}^{t-1}\right)^+$
  Observe $v^t \in \mathbb{R}^{|A|}$
  $X^t \leftarrow [] \in \mathbb{R}^{|A|}$
  $\tilde{\bar{r}}^t \leftarrow [] \in \mathbb{R}^{|A|}$ ▷ Bootstrapped average
  for $a \in A$ **do**
    $X_a^t \leftarrow \varphi(a)$
    $r_a^t \leftarrow v_a^t - \sigma^t \cdot v^t$
    $\tilde{\bar{r}}_a^{t-1} \leftarrow \frac{\tilde{R}^{t-1}}{t-1}$
    $\tilde{\bar{r}}_a^t \leftarrow \tilde{\bar{r}}_a^{t-1} + \frac{r_a^t - \tilde{\bar{r}}_a^{t-1}}{t}$
  end for
  $f^t \leftarrow \textsc{TrainRegressor}(X^t, \tilde{\bar{r}}^t)$
end for

Regression CFR

Using RRM as the base learner in CFR results in Regression CFR (RCFR). Each sequence, Ia, where $I \in \mathcal{I}$, $P(I) = i$, and $a \in A(I)$, is given a feature expansion, $\varphi(Ia)$. Whenever player i's policy, $\sigma_i^t(\cdot|I)$, at an information set is required, player i's regression function, $f_i \colon \Phi \to \mathbb{R}$, is used to generate $\tilde{R}^t$ and subsequently $\sigma^t(\cdot|I)$. Instead of independent no-regret learners each contributing their policy, $\sigma_i^t(\cdot|I)$, to the player's current strategy, a single shared regressor or an ensemble of regressors generates the entire current strategy, σ_i^t. Using a shared regressor allows generalization over distinct sequences, thereby enabling games to be solved with a compact representation that is potentially a fraction of the size of the original game.

Procedurally, each iteration of RCFR consists of two steps. The first step is essentially an iteration of CFR, with two differences. The first difference is that the strategies are generated from regressors rather than tables of cumulative regrets. The second is that instantaneous regrets that are computed during a walk of the game tree and traditionally are used to update cumulative regrets are instead added to a table of regression data along with the features of their associated game sequences. Once the CFR-like iteration is finished we can use this regression data to update or retrain a

regressor. The updated regressor is then used on the next RCFR iteration to generate the player's strategy. Just like in CFR, the average of the reach probabilities to every sequence can be tracked to yield an average strategy profile. The only difference in RCFR is that the reach probabilities are generated by querying regressors instead of tables.

Inherited from RRM, RCFR's regret bound depends on the accuracy of its regression system.

Theorem 3.0.7. *Let $N_i = \max_{I \in \mathcal{I}_i} |A(I)|$. Then RCFR bounds the external regret for player i after T iterations by*

$$|\mathcal{I}_i| \sqrt{L\sqrt{N_i}\left(LT\sqrt{N_i} + 4\sum_{t=1}^{T}\epsilon^t\right)} \tag{3.48}$$

so as long as its approximate regrets, $\tilde{R}^t(\cdot|I)$, at every information set I, on round t, has bounded ℓ_1 error, $\left\|R^t(\cdot|I) - \tilde{R}^t(\cdot|I)\right\|_1 \leq \epsilon^t$. Therefore, as long as RCFR's regression system is accurate enough that $\sum_{t=1}^{T}\left\|R^t(\cdot|I) - \tilde{R}^t(\cdot|I)\right\|_1 < T^2$ at every information set I, RCFR's average profile will converge to a Nash equilibrium.

Proof. By Corollary 3.0.5, RRM has bounded counterfactual regret on all information sets $I \in \mathcal{I}_i$,

$$R_i^T(I) \leq \sqrt{L\sqrt{N_i}\left(LT\sqrt{N_i} + 4\sum_{t=1}^{T}\epsilon^t\right)}.$$

Using the fact that $R_i^{\text{ext},T} \leq \sum_{I \in \mathcal{I}_i} R_i^{T,+}(I)$ from Zinkevich et al. [43], the theorem is proven:

$$\begin{aligned} R_i^{\text{ext},T} &\leq \sum_{I \in \mathcal{I}_i} \sqrt{L\sqrt{N_i}\left(LT\sqrt{N_i} + 4\sum_{t=1}^{T}\epsilon^t\right)} \\ &\leq |\mathcal{I}_i| \sqrt{L\sqrt{N_i}\left(LT\sqrt{N_i} + 4\sum_{t=1}^{T}\epsilon^t\right)}. \end{aligned}$$

□

Relationship to Abstraction

The regressor can be thought of as a dynamic "soft" abstraction that relates the regret values of distinct player sequences to each other, without necessarily collapsing

them together completely. Such an abstraction is soft in contrast to traditional "hard" abstraction techniques [17, 21] that must merge compatible information sets so they are indistinguishable. The complexity of the regressor, *e.g.* the number of weights in a linear model or the number of nodes in a regression tree, is analogous to the size of the abstraction. And while traditional abstractions must be set in advance of the game solving process, RCFR's game representation adapts during solving.

Action abstraction however, does not neatly fit under the RCFR umbrella. RCFR does not have a convenient way of preventing the counterfactual values of certain actions from influencing the regrets of other actions or preventing certain actions from being played, yet these are the fundamental effects of traditional action abstractions. These two effects may be useful in preventing risky actions from being played as discussed in Section 2. But RCFR computes an approximate equilibrium in the full game, thereby avoiding possible detrimental sequence removal effects. The experiments in Section 4 illustrate and explore this trade-off.

RRM$^+$ and RCFR$^+$

If negative exact or approximate average regrets, $\bar{r}_a^t$ or $\tilde{\bar{r}}_a^t$, are set to zero in Algorithm 1 or 2, the regression targets become average Q-regrets or approximate average Q-regrets, respectively. Thus the resulting algorithms are RM$^+$ variants of RRM and we denote them RRM$^+$.

Recall that negative regrets are set to zero by RM before it generates its strategy. This means that *all* non-positive regrets are mapped to the same probability mass. But a regressor in RRM that attempts to optimize squared or absolute loss would see a large loss between an estimate of 0 and a true regret of -100, for example. The RRM regressor is then encouraged to use some of its representational power to accurately estimate the precise values for negative regrets. Since negative approximate regrets will be set to zero during strategy formation, this representational power is essentially wasted. Said another way, a regressor that is only more accurate when estimating negative values will not make RRM's strategy approximation any more accurate. If instead the regressor used all of its representational power to accurately estimate

positive regrets, then its strategies would certainly better resemble those that would have been played by RM.

RRM$^+$ avoids this problem entirely because its regressor is trained to estimate Q-regrets, which are by definition non-negative. All of the regressor's representational power in this case is used to estimate values that will have an impact on the resulting strategy. Any improvement in the regressor's accuracy must improve RRM$^+$'s strategy approximation.

In addition, RRM$^+$ is theoretically sound, having regret bounds similar to those of RRM. They differ only in that the error of the estimates produced by the RRM$^+$ regressor are measured with respect to Q-regrets rather than conventional regrets. But before proving RRM$^+$'s regret bound, we must prove a variation of ϵ-Blackwell's condition with Q-regrets as targets and some elementary facts about Q-regret.

Definition 3.0.8 (Instantaneous Q-regret)**.** *Derived from the definition of cumulative Q-regret, the **instantaneous Q-regret** on iteration t is*

$$\begin{aligned} q^t &= Q^t - Q^{t-1} \\ &= (Q^{t-1} + r^t)^+ - Q^{t-1} \\ &= \begin{cases} Q^{t-1} + r^t - Q^{t-1}, & \textit{if } r^t > -Q^{t-1} \\ -Q^{t-1}, & \textit{otherwise} \end{cases} \\ &= \begin{cases} r^t, & \textit{if } r^t > -Q^{t-1} \\ -Q^{t-1}, & \textit{otherwise} \end{cases} \\ &= \max\{\, r^t, -Q^{t-1} \,\} \end{aligned}$$

where $Q^0 = 0$. Alternatively, the condition on which $q^t = -Q^{t-1}$ can be restated as

$$r^t \leq 0 \textit{ and } \left|r^t\right| \geq \left|Q^{t-1}\right| = Q^{t-1}.$$

Definition 3.0.9 (ϵ-Blackwell's Condition with Q-regret)**.** *Then Blackwell's condition is satisfied with $\epsilon \in \mathbb{R}$ error if*

$$\sup_{v^{t+1} \in \mathcal{V}} \nabla\Phi(Q^t) \cdot q^{t+1} \leq \epsilon. \tag{3.49}$$

Theorem 3.0.10 (Bound on Error in Achieving Blackwell's Condition With Respect to Q-regret)**.** *Consider the online learning setting as described in Section 2 with*

action set A. A learner that uses Method 2.1 with weights $\omega \coloneqq y^+$, where $y \in \mathbb{R}^{|A|}$, to generate her strategy on round $t+1$, satisfies ϵ-Blackwell's condition with ϵ at most

$$4 \left\| v \right\|_2 \left\| Q^t - y^+ \right\|_1 . \tag{3.50}$$

Proof. Consider arbitrary $v = v^{t+1} \in \mathcal{V}$. Expanding and doing algebra on the left-hand-side of Blackwell's condition for Q-regret,

$$\begin{aligned} \nabla \Phi(Q^t) \cdot q^{t+1} &= 2Q^{t,+} \cdot q^{t+1} \\ &= 2Q^t \cdot \begin{cases} -Q^t, & \text{if } r^{t+1} \leq -Q^t \\ r^{t+1}, & \text{otherwise.} \end{cases} \end{aligned}$$

When $Q^t = \mathbf{0}$, Blackwell's condition is satisfied exactly, so consider now only the case where $Q^t \neq \mathbf{0}$. Consider the case when $q^{t+1} = -Q^t$. Since each element of Q^t is non-negative by the definition of Q-regrets,

$$\begin{aligned} &= 2Q^t \cdot -Q^t \\ &\leq 2Q^t \cdot \mathbf{0} = 0, \end{aligned}$$

which proves the theorem for this case. In the remaining case,

$$= 2Q^t \cdot r^{t+1}.$$

Repeating the same steps as those in the proof of Theorem 3.0.3 with Q^t substituted for $R^{t,+}$ concludes the proof by yielding

$$\leq 4 \left\| v \right\|_2 \left\| Q^t - y^+ \right\|_1 .$$

□

Lemma 3.0.11. *The squared ℓ_2 norm of instantaneous regret is an upper bound on that of instantaneous Q-regret.*

Proof. Consider the instantaneous regret and Q-regret on a single action $a \in A$.

$$\begin{aligned} (q^t)^2 &= (\max\{\, r^t, -Q^{t-1} \,\})^2 \\ &= \begin{cases} (Q^{t-1})^2, & \text{if } r^t \leq 0 \text{ and } |r^t| \geq |Q^{t-1}| = Q^{t-1} \\ (r^t)^2, & \text{otherwise} \end{cases} \\ &= \begin{cases} (Q^{t-1})^2, & \text{if } r^t \leq 0 \text{ and } (r^t)^2 \geq (Q^{t-1})^2 \\ (r^t)^2, & \text{otherwise} \end{cases} \end{aligned}$$

Since $(q^t)^2 = (Q^{t-1})^2$ only when $(r^t)^2 \geq (Q^{t-1})^2$,

$$(q^t)^2 \leq (r^t)^2.$$

Since the squared ℓ_2 norm of a vector $y \in \mathbb{R}^{|A|}$, $\|y\|_2^2 = \sum_{i=1}^{|A|} y^2$,

$$\begin{aligned}\left\|q^t\right\|_2^2 &= \sum_{a \in A} (q_a^t)^2 \\ &\leq \sum_{a \in A} (r_a^t)^2 \\ &\leq \left\|r^t\right\|_2^2.\end{aligned}$$

□

Theorem 3.0.12 (Regret Bound Given ϵ^t-Blackwell's Condition on Q-regret is Satisfied)**.** *If a learner satisfies ϵ^t-Blackwell's condition with respect to Q-regret on every round t, then the learner's regret is bounded by $\sqrt{T\,|A|\,L^2 + \sum_{t=1}^{T} \epsilon^t}$. This is the same regret bound one would achieve by satisfying ϵ^t-Blackwell's condition with respect to regrets.*

Proof.

By Lipschitz continuity of the gradient of Φ,

$$\Phi(Q^{t+1}) \leq \Phi(Q^t) + \nabla\Phi(Q^t) \cdot \left(Q^{t+1} - Q^t\right) + \left\|Q^{t+1} - Q^t\right\|_2^2 \tag{3.51}$$

$$= \Phi(Q^t) + \nabla\Phi(Q^t) \cdot q^{t+1} + \left\|q^{t+1}\right\|_2^2. \tag{3.52}$$

By ϵ^{t+1}-Blackwell's condition and Lemma 3.0.11,

$$\leq \Phi(Q^t) + \epsilon^{t+1} + \left\|r^{t+1}\right\|_2^2 \tag{3.53}$$

$$\leq (t+1)\,|A|\,L^2 + \sum_{s=1}^{t+1} \epsilon^s. \tag{3.54}$$

Bounding the maximum regret in terms of $\Phi(Q^T)$, we find

$$\max_{a \in A} R_a^T \leq \max_{a \in A} Q_a^T, \text{ by Tammelin et al.'s Lemma 1 [38].} \tag{3.55}$$

$$= \left\|Q^T\right\|_\infty \leq \left\|Q^T\right\|_2 = \sqrt{\Phi(Q^T)} \tag{3.56}$$

$$\leq \sqrt{T\,|A|\,L^2 + \sum_{t=1}^{T} \epsilon^t}. \tag{3.57}$$

□

Theorems 3.0.10 and 3.0.12 together show that measuring the deviation from regrets and Q-regrets yields the same regret bound, so RRM$^+$ has the same regret bound as RRM except that their estimation targets differ. As an aside, note that just as Theorem 3.0.5 is an alternative proof of RM's regret bound, Theorem 3.0.12 is an alternative proof of RM$^+$'s external regret bound.

Just as RRM can be used in CFR in place of RM, so can RRM$^+$. We call the variant of RCFR that uses RRM$^+$ RCFR$^+$. Procedurally, RCFR$^+$ is no different from RCFR, except that Q-regrets or approximate Q-regrets are used as regression targets. RCFR$^+$ also retains the same guarantees as RCFR by combining the bound of Corollary 3.0.5, and Theorems 3.0.12 and 3.0.7.

Chapter 4

Experiments

We compare the exploitability of strategies computed by RCFR variants against each other and abstracted CFR with similarly complex game representations, to illustrate the empirical advantage of RCFR and explore some choices in its application. We compare RCFR's performance in Leduc hold'em against traditional state-space abstractions, and in no-limit one-card poker against traditional action abstractions, as well as unabstracted CFR$^+$ in both games. The iterative behaviour of some of these RCFR and CFR variants is shown to qualitatively compare their convergence. 100,000 solver iterations were used to generate all strategies. The traditional abstract games were solved with CFR$^+$[1].

The following RCFR variants were tested:

- RCFR with a single best-first regression tree for each player and stored $\bar{r}^t$ RRM (BF).
- RCFR with a single best-first regression tree for each player and bootstrapped RRM ($\texttt{BF}_{\texttt{BS}}$).
- RCFR$^+$ with a single best-first regression tree for each player and stored $\bar{q}^t = \frac{Q^t}{t}$ RRM$^+$ ($\texttt{BF}^+$).
- RCFR$^+$ with a single best-first regression tree for each player and bootstrapped RRM$^+$ ($\texttt{BF}^+_{\texttt{BS}}$).

In addition, we also compare these new RCFR variants with the results for RCFR with a single recursive partitioning regression tree and stored $\bar{r}^t$ RRM (RP) that were

[1] This setup deviates from Waugh et al.'s [41] experiments where traditional abstract games were solved with chance-sampled CFR [44], but the exploitabilities of the resulting strategies are similar.

originally presented with Waugh et al. [41]. Recursive partitioning regression tree RCFR versions were not run in no-limit one-card poker due to the lack of a clear advantage in Leduc over the analogous best-first version, `BF`, and the significant tuning required to find error tolerances that match desired regressor complexities. All regression trees minimized squared training error. While the theory presented in Chapter 3 states that minimizing absolute error yields a regret bound where regression errors are not scaled by an additional $\sqrt{|A|}$ factor, this was discovered after experiments were completed. Prior to this new theory, minimizing squared error was believed to yield the best regret bound.

Features

The feature expansion we used in both games for best-first regression tree RCFR variants[2] is the concatenation of a card feature expansion and a betting feature expansion. To accommodate no-limit histories, the betting feature expansion is itself a concatenation of a vector of features for each information set–action pair in the history's action sequence, listed from most to least recent. Each feature is an elementary descriptor of a poker sequence component, and is meant to be extendable to larger poker games with little modification.

- **Card Features**
  - The acting player's expected hand strength ($E[HS]$), which is the probability of winning the hand given the available information, and marginalized over a uniform distribution of opponent hands and the possible future board card.
  - The rank of the board card or zero on the preflop.
- **Information Set-Action Features**
  - Feature that indicates player 1 is next to act.
  - Feature that indicates player 2 is next to act.

[2]The recursive partitioning tree used alternative betting features that do not extend well from limit to no-limit poker. They are described by Waugh et al. [41].

- The pot size after this action.
- The acting player's remaining stack size.
- Feature that indicates the action was a fold.
- Feature that indicates the action was a call.
- Feature that indicates the action was a non-all-in wager.
- Feature that indicates the action was an all-in wager.
- The wager size in chips.
- The wager size in fractions of the pot after matching the opponent's current contribution.
- The wager size in fractions of the acting player's remaining stack.

Static Abstractions to Compare with RCFR

Here we describe the design of the traditional static abstractions that we compare with RCFR in our two chosen poker games. Each type of abstraction was chosen as a natural abstraction in their respective game. For Leduc hold'em, this is state-space abstraction that obscures the exact identity of the private and public cards, while in no-limit one-card poker, this is a pot fraction action abstraction where a small number of different pot fraction sized wagers are allowed. The range of abstraction sizes was chosen to reflect the typical range of abstraction sizes for each abstraction type. Each abstraction has been used in previous works in the same or similar testbeds [42, 40, 31].

Leduc Hold'em

In Leduc hold'em, a typical abstraction is one that groups together cards on the preflop and only distinguishes between pairing and non-pairing board cards on the flop. We use the abstractions defined by Waugh et al. [42], which are hand-crafted abstractions that are analogous to the $E[HS]$ based abstractions commonly used in Texas hold'em. Abstractions are denoted, for example, `J.QK` to describe the abstraction that can distinguish between a jack and a queen or king, but cannot distinguish between a queen and king. The remaining three abstractions are then

JQK, JQ.K, and J.Q.K. One may also note that J.Q.K is a strict refinement of the other three abstractions, and J.QK and JQ.K both are strict refinements of JQK.

No-limit One-card Poker

A typical poker action abstraction is one that declares a set of wagers, usually in the form of pot size fractions, that the agents are allowed to play along with check/call, fold, and all-in [16, 42, 31]. All other wagers are disallowed.

In order to play in the unabstracted game, actions played in the real game must be translated into the abstract game, and action choices by the abstract game solution must be translated into the real game. While there is currently no clear best translation scheme, we follow Schnizlein et al.'s [31] advice and use abstract, soft, geometric translation, where a wager is never interpreted as a check/call or fold. "Abstract" refers to how actions are translated based on the pot size in the abstract game rather than the real game. "Soft" signifies that each potential translation of a real game action in a sequence is given a weight and the weight is sampled (consistently within the hand) to obtain a single abstract sequence. And "geometric" refers to the geometric similarity metric introduced by Gilpin et al. [16]. This configuration is typically competitive with alternatives in terms of exploitability, and it is still used in the ACPC agents fielded by the University of Alberta's Computer Poker Research Group.

We denote allowed wagers by a single letter code where H is half the pot size, P is the pot size, and D is double the pot size. We also use $\emptyset$ to denote the null betting abstraction where only check/call, fold, and all-in are allowed. $\emptyset$, P, HP, PD, and HPD abstractions are used as our action abstraction exemplars in no-limit one-card poker.

Results

Comparisons of the final exploitabilities of profiles found with different methods are shown in Figures 4.1 and 4.3. They are aligned by complexity on the horizontal axis as a percentage of the size of the unabstracted game, so points along the same vertical line were found with approaches requiring a similar complexity. RCFR's

complexity is the size of its regression trees while the complexity of a conventional abstraction is the size of its abstract game. The subfigures with a linear vertical axis readily shows the raw performance of each of the techniques, while the subfigures with a logarithmic vertical axis reflect the increasing computational difficulty of reducing exploitability as exploitability decreases.

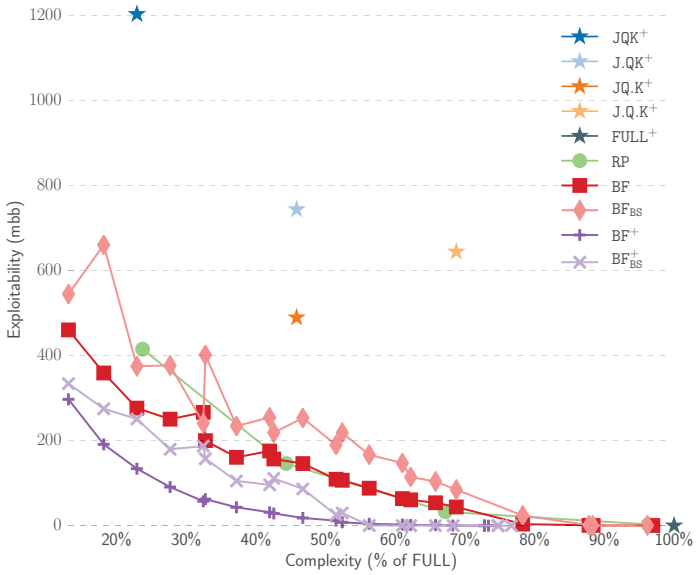
Convergence graphs, Figures 4.2 and 4.4, show the progress of some of these strategies during their solver iterations. Each of the methods within a convergence graph have a similar size, except for unabstracted $\texttt{FULL}^+$, which is included for comparison.

Analysis

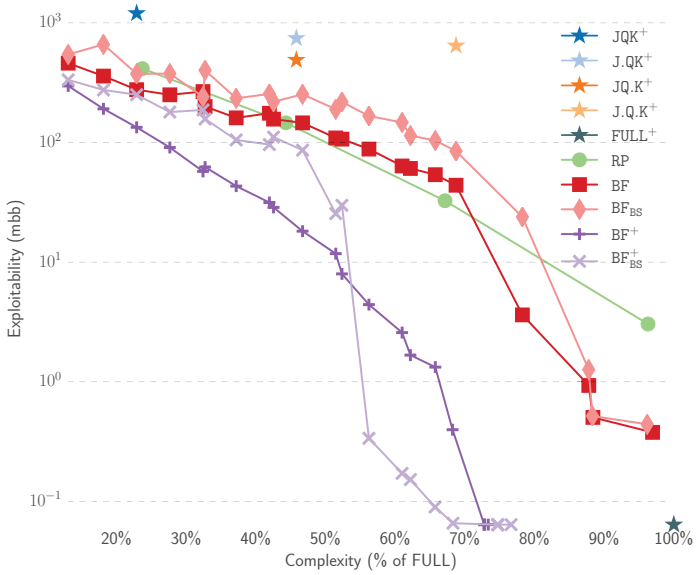
BF and RP perform similarly well in Leduc hold'em, as shown by Figure 4.1. The lower exploitability of BF near full game size is due to BF using alternating CFR updates rather than RP's simultaneous updates, since both are essentially tabular here. This suggests that the best-first regression tree, with its greater simplicity in terms of enforcing a complexity budget is a better choice for use in RCFR.

RCFR$^+$ outperforms plain RCFR at every complexity level in both games, and dramatically so at moderate complexities. Since all points besides those very near to full game complexity, including those using plain RCFR have largely converged, this is not due to RM$^+$'s better empirical convergence properties. Instead, this shows that in these games Q-regrets are significantly easier for regression trees to learn than regrets. In fact, $\texttt{BF}^+$ and $\texttt{BF}^+_{\texttt{BS}}$ were run with complexity budgets larger than 80% of Leduc hold'em, but Figure 4.1 shows that less than 80% of the game size was all that was required to perfectly estimate the Q-regrets. And when the Q-regrets were estimated perfectly, or nearly so, RCFR$^+$ matched the convergence performance of $\texttt{FULL}^+$, so RCFR$^+$ also inherits the improved convergence properties that have been observed for CFR$^+$.

Every version of RCFR outperforms every state-space abstraction in Leduc hold'em (see Figure 4.1) by a wide margin. The action abstractions in one-card poker are more effective (see Figure 4.3), but both RCFR$^+$ variants outperform all



(a) Linear scale.



(b) Logarithmic scale.

Figure 4.1: Exploitability of the final average strategies of RCFR and CFR variants in Leduc hold'em.

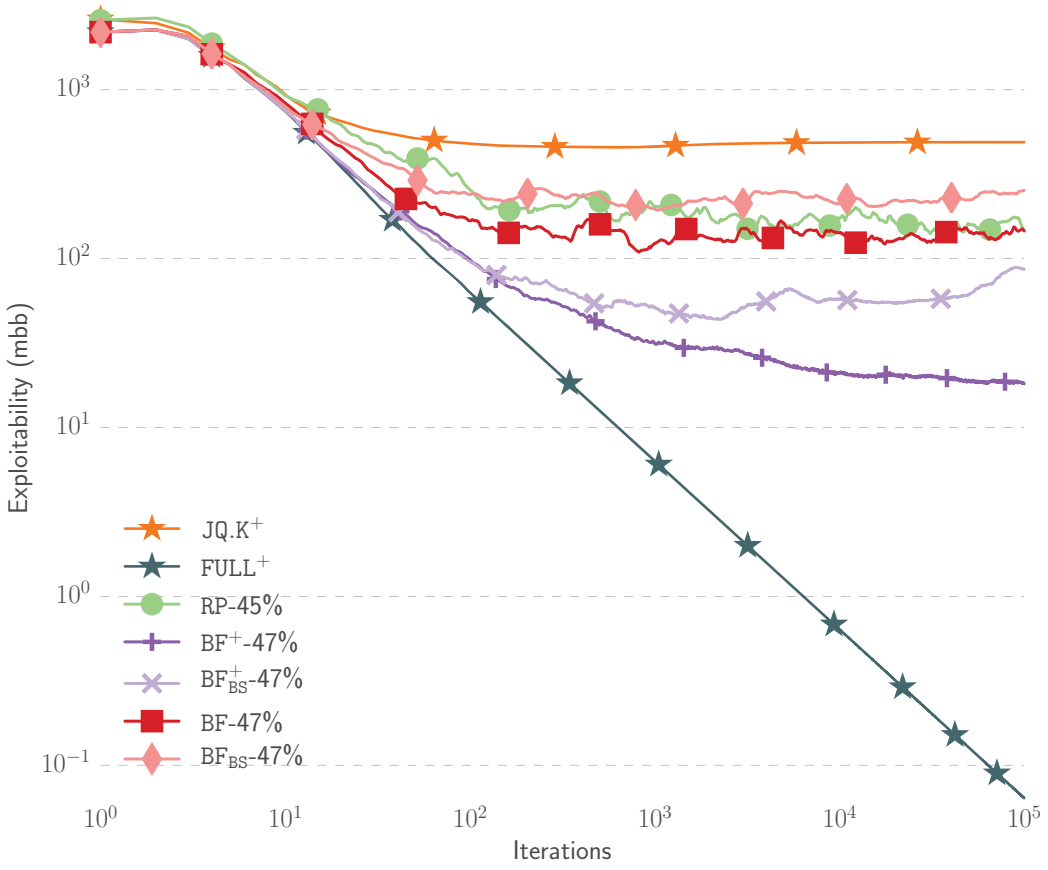
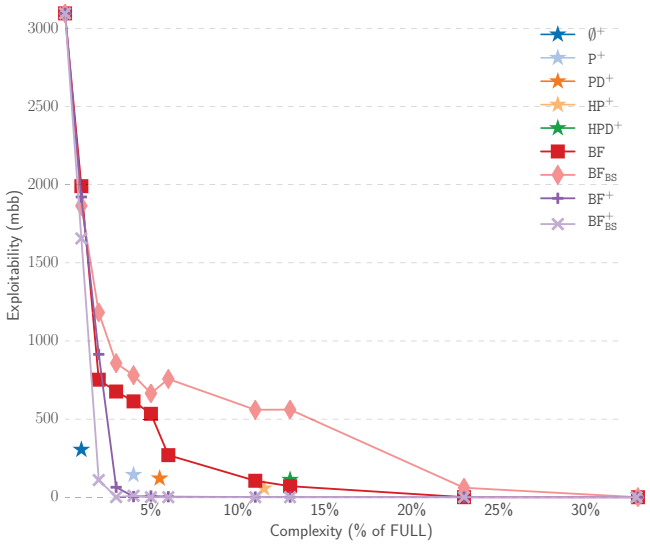
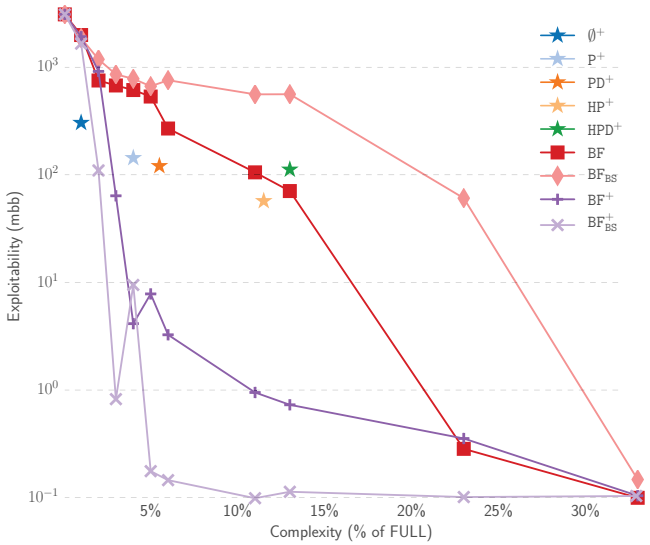


Figure 4.2: Convergence of CFR variants that use representations that are near 45% of Leduc hold'em's size, along with unabstracted FULL$^+$ for comparison.



(a) Linear scale.



(b) Logarithmic scale.

Figure 4.3: Exploitability of the final average strategies of RCFR and CFR variants in no-limit one-card poker. Runs with more than 35% of the game's complexity have been omitted because they all achieved the same performance as FULL$^+$.

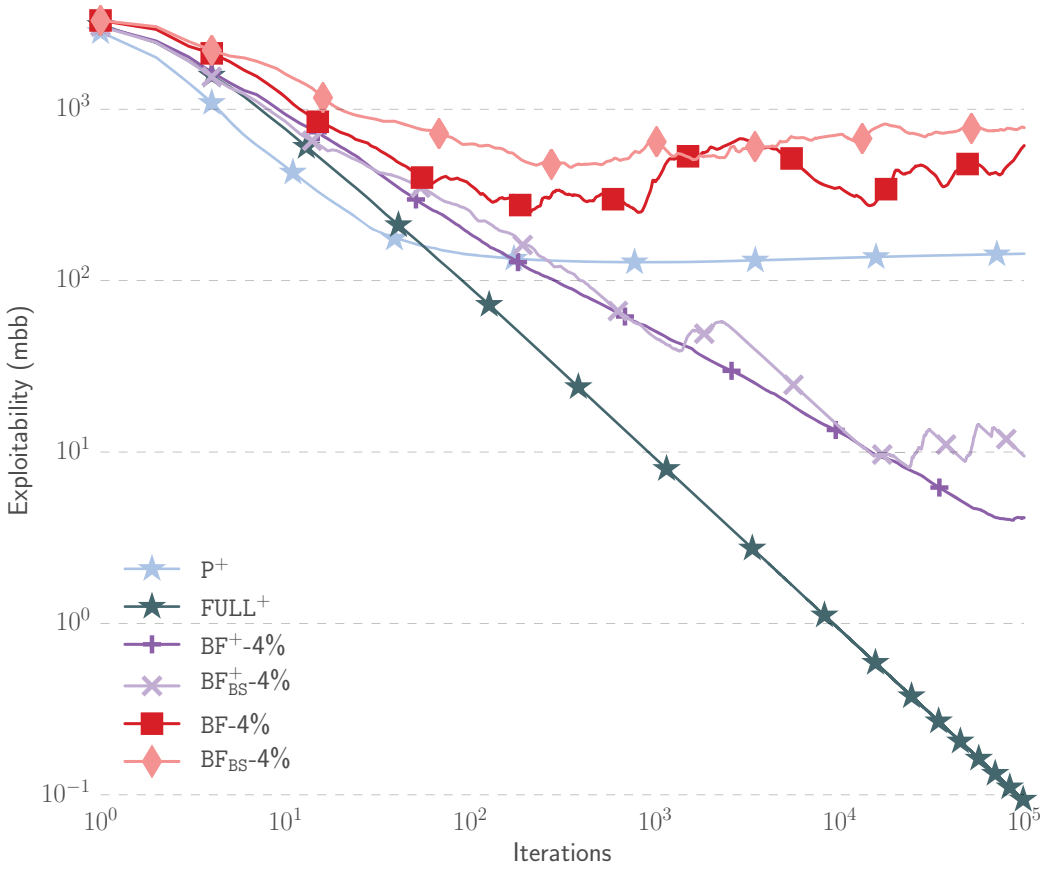


Figure 4.4: Convergence of CFR variants that use representations that are near 4% of one-card poker's size, along with unabstracted FULL$^+$ for comparison.

but the null abstraction at its complexity. Even so, RCFR$^+$ outperforms even the null abstraction with only 2% greater complexity. As discussed in Section 2, action abstractions that limit the number of risky options, as $\emptyset$ does, tend to be economical in terms of the exploitability they reach versus the abstract game's size. But even the abstract game with one non-all-in wager, `P`, is complex enough to make RCFR$^+$ the clearly better approach.

As expected, bootstrapping does tend to decrease RCFR's performance. Fortunately, the estimation errors do not appear to be compounding wildly either, as the bootstrapped versions of RCFR still perform near their non-bootstrapped counterparts, and outperform many of the static abstractions. In addition, the bootstrapped version of RCFR$^+$, $\texttt{BF}^+_{\texttt{BS}}$, outperforms all other variants at almost every complexity level in one-card poker. Thus, using bootstrapping with RCFR$^+$ may be an effective way of reducing the storage requirements of RCFR without severely increasing the exploitability of the final strategy profile.

Another trend to note is that RCFR largely avoids abstraction pathologies. For the most part, the lines in Figures 4.1 and 4.3 approach zero as complexity increases. However, pathologies do occur, particularly in bootstrapped versions, as one might expect from the estimation error added by bootstrapping.

Chapter 5

Conclusions

In this thesis, we introduced RCFR, an imperfect information game solving technique that can compactly solve large games without abstracting them *a priori*. RCFR is an alternative to both state-space and action abstraction methods to approximately solve large games with compact representations. We showed that RCFR and its variants are theoretically sound, with regret and equilibrium guarantees. RCFR variants allow practitioners to avoid the challenge of creating abstractions, which requires preparation and domain knowledge before the solving process can begin. As a side-effect, RCFR avoids sequence removal effects that can arise in imperfect information games. These effects are liable to trip-up abstraction designers and lead to strategy profiles that are highly exploitable in the real game. Furthermore, its compact representation is shaped by the data it obtains while solving the game, making it a powerful and flexible dynamic abstraction technique.

In experiments, RCFR tends to produce less exploitable strategies than its traditional abstraction counterparts. RCFR$^+$, even with bootstrapped Q-regret estimates, generates strategy profiles with similar complexities but lower exploitabilities than traditional abstractions in two poker games: Leduc hold'em, a small limit variant of Texas hold'em, and no-limit one card poker. In addition, both RCFR and RCFR$^+$ variants solve these games nearly as well as unabstracted CFR$^+$ with representations that are a fraction of the size.

Future Work

The games solved by RCFR variants in this thesis are still small, so the next obvious step is to scale to larger games that *require* some form of abstraction, such as no-limit Texas hold'em or phantom hex on a large board. The following problems remain to be investigated and solved to achieve this goal.

Various sampling schemes that contribute heavily to CFR's popularity have yet to be incorporated into RCFR. Such extensions should be straightforward, both for RCFR's theory and implementation, but we are concerned about how the sampling variance and regressor error will interact. We suspect that lower variance sampling schemes, perhaps similar to the probing method proposed by Gibson et al. [15], that limit such interaction will perform better. This would particularly be a problem for RCFR$^+$, as CFR$^+$ with almost any sampling tends to perform worse than plain CFR with the same sampling scheme. It may be that if sampling is necessary, one would be better off using a different work-around to make the regression problem better reflect the strategy generation problem, such as estimating the probability mass for each action directly, rather than estimating regrets or Q-regrets. This could be done by extending exponential weights/hedge [11] to regression hedge. In addition, adapting the DAGGer algorithm [30] to extensive-form games and using it in the context of RCFR could reduce the impact of regression errors on the exploitability of the resulting strategy profile by aligning the regressor's strategy with the sequence distribution imposed by the observed regrets.

In our experiments, the average strategy was stored exactly as a table, which would be infeasible for larger games. This could be avoided by using another regressor instead of a table, but this would introduce another potential compounding error problem. DAGGer [30] might be a fix for such a problem, but an alternative is to save a copy of RCFR's regressors that generate the current strategies for each player every $n > 1$ iterations. Rather than storing the average explicitly, it could be computed on demand by querying the (compact) current strategy checkpoints. Another alternative for RCFR$^+$ is to forgo computing the average at all, since it has been observed that the current strategies of RM$^+$-based solvers tend to converge to

zero exploitability as well [37, 6, 38].

RCFR still requires some domain knowledge in the form of an adequate feature representation, but this could potentially be learned as well. The regression trees employed for the experiments in this thesis do automatically generate a type of feature representation in the form of a training data partitioning. More sophisticated representation learning strategies such as deep neural networks and autoencoders [3] might be effective as well if they could be run fast enough. Or perhaps such techniques could be used offline to generate an effective "base" representation to facilitate the learning of a linear function online. Since many RCFR iterations are required for convergence, whatever regressor is used must be fast, both to train and to query.

Bibliography

[1] Baruch Awerbuch and Robert D. Kleinberg. "Adaptive routing with end-to-end feedback: distributed learning and geometric approaches". In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. 2004, pp. 45–53. DOI: 10.1145/1007352.1007367. URL: http://doi.acm.org/10.1145/1007352.1007367.

[2] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. "Online Implicit Agent Modelling". In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2013.

[3] Yoshua Bengio, Aaron Courville, and Pierre Vincent. "Representation learning: A review and new perspectives". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8 (2013), pp. 1798–1828.

[4] Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. "Approximating game-theoretic optimal strategies for full-scale poker". In: *Proceedings of the 18th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc. 2003, pp. 661–668.

[5] David Blackwell. "An analog of the minimax theorem for vector payoffs." In: *Pacific J. Math.* 6.1 (1956), pp. 1–8. URL: http://projecteuclid.org/euclid.pjm/1103044235.

[6] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. "Heads-up limit hold'em poker is solved". In: *Science* 347.6218 (2015), pp. 145–149.

[7] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN: 0-534-98053-8.

[8] Neil Burch, Marc Lanctot, Duane Szafron, and Richard Gibson. "Efficient Monte Carlo counterfactual regret minimization in games with many player actions". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1880–1888.

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

[10] Rémi Coulom. "Computing Elo Ratings of Move Patterns in the Game of Go". In: *International Computer Games Association Journal* 30 (2007), pp. 198–208.

[11] Yoav Freund and Robert E Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “Additive Logistic Regression: A Statistical View of Boosting”. In: *Annals of Statistics* (2000), pp. 337–374.

[13] Sam Ganzfried and Tuomas Sandholm. “Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping”. In: *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press. 2013, pp. 120–128.

[14] Sylvain Gelly and David Silver. “Achieving Master Level Play in 9 x 9 Computer Go.” In: *23rd Conference on Artificial Intelligence*. 2008, pp. 1537–1540.

[15] Richard Gibson, Marc Lanctot, Neil Burch, Duane Szafron, and Michael Bowling. “Generalized Sampling and Variance in Counterfactual Regret Minimization”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[16] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. “A heads-up no-limit Texas Hold’em poker player: discretized betting models and automatically generated equilibrium-finding programs”. In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems. 2008, pp. 911–918.

[17] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. “Potential-aware Automated Abstraction of Sequential Games, and Holistic Equilibrium Analysis of Texas Hold’em Poker”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2007.

[18] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. “Logarithmic Regret Algorithms for Online Convex Optimization”. In: *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*. 2006, pp. 499–513. DOI: 10.1007/11776420_37. URL: http://dx.doi.org/10.1007/11776420_37.

[19] Shih-Chieh Huang, Broderick Arneson, Ryan B Hayward, Martin Müller, and Jakub Pawlewicz. “MoHex 2.0: a pattern-based MCTS Hex player”. In: *Computers and Games*. Springer, 2014, pp. 60–71.

[20] Laurent Hyafil and Ronald L. Rivest. “Constructing Optimal Binary Decision Trees is NP-Complete”. In: *Inf. Process. Lett.* 5.1 (1976), pp. 15–17. DOI: 10.1016/0020-0190(76)90095-8. URL: http://dx.doi.org/10.1016/0020-0190(76)90095-8.

[21] M. Johanson. “Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player”. MA thesis. University of Alberta, 2007.

[22] Michael Johanson. *Measuring the Size of Large No-Limit Poker Games*. Technical Report TR13-01. Department of Computing Science, University of Alberta, 2013.

[23] Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. "Finding Optimal Abstract Strategies in Extensive Form Games". In: *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI)*. 2012.

[24] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. "Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization". In: *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 2012.

[25] Michael Johanson, Michael Bowling, Kevin Waugh, and Martin Zinkevich. "Accelerating Best Response Calculation in Large Extensive Games". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*. 2011, pp. 258–265.

[26] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. "Evaluating state-space abstractions in extensive-form games". In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2013, pp. 271–278.

[27] Marc Lanctot, Neil Burch, Martin Zinkevich, Michael Bowling, and Richard G Gibson. "No-Regret Learning in Extensive-Form Games with Imperfect Recall". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. 2012, pp. 65–72.

[28] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. "Monte Carlo sampling for regret minimization in extensive games". In: *Advances in Neural Information Processing Systems*. 2009, pp. 1078–1086.

[29] Martin J. Osborne and Ariel Rubinstein. *A Course On Game Theory*. MIT Press, 1994.

[30] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[31] David Schnizlein, Michael Bowling, and Duane Szafron. "Probabilistic state translation in extensive games with large action sets". In: *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*. 2009, pp. 276–284.

[32] Haijian Shi. "Best-first decision tree learning". MA thesis. University of Waikato, 2007.

[33] Jiefu Shi and Michael L. Littman. “Abstraction Methods for Game Theoretic Poker”. In: *Computers and Games, Second International Conference, CG 2000, Hamamatsu, Japan, October 26-28, 2000, Revised Papers*. 2000, pp. 333–345. DOI: 10.1007/3-540-45579-5_22. URL: http://dx.doi.org/10.1007/3-540-45579-5_22.

[34] Finnegan Southey, Michael H. Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and D. Chris Rayner. “Bayes’ Bluff: Opponent Modelling in Poker”. In: *UAI ’05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*. 2005, pp. 550–558. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1216&proceeding_id=21.

[35] Finnegan Southey, Bret Hoehn, and Robert C. Holte. “Effective short-term opponent exploitation in simplified poker”. In: *Machine Learning* 74.2 (2009), pp. 159–189. DOI: 10.1007/s10994-008-5091-5. URL: http://dx.doi.org/10.1007/s10994-008-5091-5.

[36] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, 1998.

[37] Oskari Tammelin. “Solving Large Imperfect Information Games Using CFR+”. In: *arXiv preprint arXiv:1407.5042* (2014).

[38] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. “Solving Heads-up Limit Texas Hold’em”. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 2015.

[39] Luís Torgo. “Inductive learning to tree-based regression models”. PhD thesis. University of Porto, 1999.

[40] Kevin Waugh, Nolan Bard, and Michael H. Bowling. “Strategy Grafting in Extensive Games”. In: *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.* 2009, pp. 2026–2034. URL: http://papers.nips.cc/paper/3634-strategy-grafting-in-extensive-games.

[41] Kevin Waugh, Dustin Morrill, J. Andrew Bagnell, and Michael Bowling. “Solving Games with Functional Regret Estimation”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-29, 2015, Austin Texas, USA*. Austin Texas, USA, Jan. 2015, pp. 2138–2145.

[42] Kevin Waugh, David Schnizlein, Michael H. Bowling, and Duane Szafron. “Abstraction pathologies in extensive games”. In: *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 2*. 2009, pp. 781–788. DOI: 10.1145/1558109.1558119. URL: http://doi.acm.org/10.1145/1558109.1558119.

[43] Martin Zinkevich, Michael Johanson, Michael H. Bowling, and Carmelo Piccione. “Regret Minimization in Games with Incomplete Information”. In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. 2007, pp. 1729–1736. URL: http://papers.nips.cc/paper/3306-regret-minimization-in-games-with-incomplete-information.

[44] Martin Zinkevich, Michael Johanson, Michael H. Bowling, and Carmelo Piccione. *Regret Minimization in Games with Incomplete Information*. Tech. rep. TR07-14. Department of Computing Science, University of Alberta, 2007.

Appendix A

Counterexample to Inequality 19 in Waugh et al.'s [41] Blackwell's Condition Error Bound Proof

Waugh et al. [41] presents the first proof of a bound on the error of Blackwell's condition during online learning in terms of the Euclidean distance between the learner's regrets and the weights it will play on the next round. Here we show that the proof is incorrect.

Theorem A.0.13. *Inequality 19 in the Blackwell's condition error bound proof provided by Waugh et al. [41] is false.*

Proof. To map the notation from that proof to the notation used in this thesis, note that $e = \mathbf{1}$, $(x)_+ = x^+$ for any $x \in \mathbb{R}^d$ and $d > 0$, and $\left(\tilde{R}^T\right)_+ = y$, where $y \in \mathbb{R}^{|A|}$ is an arbitrary vector of weights that the learner will use to generate its strategy in the next round.

First, we convert the error term in the right-hand-side of line 18 in Waugh et al.'s proof into our notation,

$$\left\|\left(R^T\right)_+ - \left(\left(\tilde{R}^T\right)_+ / e \cdot \left(\tilde{R}^T\right)_+\right) e \cdot \left(R^T\right)_+\right\|_2 = \left\|R^{T,+} - \left(y^+ / \mathbf{1} \cdot y^+\right) \mathbf{1} \cdot R^{T,+}\right\|_2 \tag{A.1}$$

$$= \left\| R^{T,+} - \frac{\left\|R^{T,+}\right\|_1}{\left\|y^+\right\|_1} y^+ \right\|_2 . \tag{A.2}$$

Next, we show the inequality that Waugh et al. asserts, still in our notation,

$$\left\| R^{T,+} - \frac{\left\| R^{T,+} \right\|_1}{\left\| y^+ \right\|_1} y^+ \right\|_2 \leq \left\| R^{T,+} - y^+ \right\|_2 . \tag{A.3}$$

Finally, we convert this back into the error term in the right-hand-side of line 19 in Waugh et al.'s proof,

$$\left\| R^{T,+} - y^+ \right\|_2 = \left\| \left(R^T \right)_+ - \left(\tilde{R}^T \right)_+ \right\|_2 . \tag{A.4}$$

Showing that Inequality A.3 is false would then break the connection between A.1 and A.4, thereby showing that Inequality 19 of the original proof is also false. Waugh et al. [41] justify Inequality 19 by mentioning that "projections preserve distances", but the following is a counterexample. Choose

$$\begin{aligned} R^T &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ y &= \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix} . \end{aligned}$$

Consequently,

$$\begin{aligned} R^{T,+} &= R^T \\ y^+ &= y \\ \left\| R^{T,+} \right\|_1 &= 1 \\ \left\| y^+ \right\|_1 &= \frac{1}{2} \\ \frac{\left\| R^{T,+} \right\|_1}{\left\| y^+ \right\|_1} &= \frac{1}{1/2} = 2. \end{aligned}$$

Evaluating the left-hand-side of Inequality A.3,

$$\begin{aligned} \left\| R^{T,+} - \frac{\left\| R^{T,+} \right\|_1}{\left\| y^+ \right\|_1} y^+ \right\|_2 &= \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 2 \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix} \right\|_2 \\ &= \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\|_2 \\ &= \left\| \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\|_2 \\ &= \sqrt{2} \approx 1.41. \end{aligned}$$

The right-hand-side is

$$\begin{aligned}\left\|R^{T,+} - y^{+}\right\|_2 &= \left\|\begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0\\\frac{1}{2}\end{pmatrix}\right\|_2 \\ &= \left\|\begin{pmatrix}1\\\frac{-1}{2}\end{pmatrix}\right\|_2 \\ &= \sqrt{1+\frac{1}{4}} = \sqrt{1.25} \approx 1.12.\end{aligned}$$

Since $\sqrt{2} > \sqrt{1.25}$, inequality A.3 is false, implying that Inequality 19 of the original proof is also false. □