EECS 545, Machine Learning
Fall 2009, University of Michigan, Ann Arbor

Abram Booth, Lee Callender, Craig W. Wright
Date: December 17, 2009

Construction and Evaluation of an Agent for Three Player Poker

1 Introduction

The game of poker presents a difficult and exciting challenge for the artificial intelligence community. Specifically a successful poker playing agent must act in a competitive multi-agent environment that is both stochastic and partially observable. In addition, an ideal agent should be able to manage risk as well as both model and deceive its opponents.[1]

Recent poker research has focused on the game of Texas Hold'em, with most of the research revolving strictly around the more tractable heads-up (two-player) game. [2][3][4] Unfortunately, *multiplayer* games ($n > 2$ players) suffer from an exponential blow-up in game size over their heads-up counterpart, and common methods to analytically solve two-player games are not applicable in the multiplayer domain. Coincidentally, solving an n-player zero-sum game is PPAD-complete with respect to the number of states, so approximation methods are required in order to find effective solutions to such games.[6]

We investigate an iterated reinforcement learning approach in the search for an approximate Nash equilibrium in the game of three-player limit Texas Hold'em. This approach couples *empirical game-theoretic analysis*(EGTA) with *reinforcement learning*(RL) [5] as a general methodology for approximating large games. In addition, we explore a method of opponent modeling that employees kernel density estimates to model the betting behavior of a player.

The next section of this paper briefly introduces some terms, both from poker and empirical game theory, that will be useful for understanding subsequent sections. These later sections then describe the details of our empirical game, the reinforcement learning problem, and our opponent modeling efforts. We then conclude and consider paths for future research.

2 Background and Definitions

2.1 Poker

As previously mentioned our agents have been designed to compete in limit three-player Texas Hold'em. *Limit*, in this context, implies that the size of all bets are fixed and that in each betting round there can be at most four raises.

The entire game is composed of four rounds of play called the *preflop*, *flop*, *turn*, and *river*. To begin, each player is dealt two cards face down (their hole cards) and a round of betting commences. This is called the *preflop*. The *flop* starts with the dealer dealing three cards face up on the table, and then another round of betting occurs. The cards that are dealt onto the table are referred to as *the board* and are used by all players in combination with their respective hole cards to make their final hands.

To begin the *turn* the dealer deals a single card face up and another round of betting commences. The hand concludes with the *river*, which is identical in play to the *turn* except that after the round of betting all of

the players that remain in the game must show their hand. All of the hands are then compared and the winning hand receives the bets that have been made by all players during the hand.[7]

Several metrics exist for evaluating the quality of a particular set of hole cards. One of these, called hand strength (HS), is simply the percentile ranking of a pair of hole cards given the current board. This number is computed for a particular hand by enumerating all other possible hands, and then comparing the hand in question to each of the enumerated hands. The strength of the hand in question is then indicated by the percent of hands that it would beat.

Expected hand strength (E[HS]), which we only compute for the *flop* and the *turn* is similar to HS, but it considers *all possible future outcomes* of the game to determine a ranking. In fact, E[HS] can be computed for the *preflop*, but because the calculation is so expensive it must be done off line and the results stored in a table for use during game play. We have chosen to ignore *preflop* E[HS]. On the other hand, E[HS] can not even be defined for the *river* because no more cards will be dealt.

To calculate E[HS] for pair of hole cards we consider the current state of the board, and then enumerate all possible future boards. Then for each board all possible opposing hands are enumerated. E[HS] is defined by the percentage of hands that the hand in question wins over all enumerations. To illustrate consider that $\binom{47}{2} \times (\binom{45}{2} + 1)$ hands must be evaluated to calculate this metric for the *flop*

The final metric that we use for measuring the quality of the hole cards is the value of expected hand strength squared, ($E[HS^2]$). Like HS and E[HS], this metric takes on values in the range $[0, 1]$. The contrast between E[HS] and $E[HS^2]$ is as follows. Consider a hand that through two enumerations of the board has been assigned the HS values 0.1 and 0.8. In this case $E[HS] = (0.1 + 0.8)/2$, while $E[HS^2] = (0.1^2 + 0.8^2)/2$. [8]

Agent performance is determined by measuring the number of small bets per hand (sb/h) an agent earns over the course of a match. To understand this metric consider that in our limit game the small *blind* is equal to one unit, a small bet is equal to two units, and a big bet is equal to four units. Therefore if a player always folds he will earn negative 0.5 sb/h.

2.2 Empirical Games

A **strategic game** $\Gamma = \langle N, (S_i), (u_i) \rangle$ consists of a finite set of players N, indexed by i; a non-empty set of strategies S_i for each player; and a utility function $u_i : \times_{j \in N} S_j \to \mathbb{R}$.

The **payoff** to a player i for playing strategy $s_i \in S_i$ with opponents playing joint strategy $s_{-i} \in \times_{j \neq i} S_j$ is given by $u_i(s_i, s_{-i})$. Here, the tuple $(s_i, s_{-i}) = s \in S$ is a **profile**. A **symmetric game** satisfies $S_i = S_j$ and $u_i(\cdot) = u_j(\cdot)$ for all $i, j \in N$.

Let $\triangle(\cdot)$ represent the probability simplex over a set. A **mixed strategy** σ_i is a probability distribution over strategies in S_i, with $\sigma_i(s_i)$ denoting the probability player i will play s_i. A mixed profile, σ, denotes a collection of mixed strategies. The **mixed strategy space** for player i is given by $\triangle_i = \triangle(S_i)$.

The strategy s_i' is a **best-response** for agent i with respect to a given profile s if, for all $s_i'' \in S_i$: $u_i(s_i', s_{-i}) \geq u_i(s_i'', s_{-i})$. In other words, a best-response is a strategy that will give the best possible payoff against a joint strategy s_{-i}.

A **Nash equilibrium** is a strategy profile σ where no player can increase their utility by unilaterally changing their strategy: $u_i(\sigma) \geq \max_{\sigma_i' \in \triangle_i} u_i(\sigma_i', \sigma_{-i})$ for all $i \in N$. All games must have at least one Nash equilibria and each equilibrium acts as a solution to the given game. Note that a Nash equilibrium is also defined as a profile in which each strategy σ_i is a best-response to the joint strategy σ_{-i} for all $i \in N$.[**?**]

For a given game, if the utility function u_i isn't known, an **empirical game** $\hat{\Gamma} = \langle N, (S_i), (\hat{u}_i) \rangle$ can be created to estimate $\hat{u}_i$ for the true utility function u_i. This is most commonly done through monte-carlo sampling over all profiles in the strategy space.

As it will be useful in the next section, let $\Gamma_{S\downarrow X}$ be a **restricted game** of the **full game** Γ, where each player is restricted to playing strategies in $X_i \subseteq S_i$.

3 EGTA/RL

In the general EGTA/RL methodology, we evaluate and solve restricted empirical games coupled a careful selection of strategies over the strategy space. New strategies are generated in response to equilibrium profiles of the restricted game of currently known candidate strategies. In searching for Nash equilibria of any game, we need to understand the relative performance of a strategy over a space of other profiles of strategies. Unfortunately, for games with very large (or possibly infinite) strategy spaces, there are insufficient computational resources available to evaluate a strategy over the entire strategy space of the game. In the *empirical game-theoretic analysis*(EGTA) approach [**?**] games are estimated through evaluating small subsets of known candidate strategies that aim to cover a broad portion of the real strategy space.

We must be selective of adding new strategies to the preexisting set of candidates, as each new strategy exponentially increases the size of the profile space that we have to sample (as we are estimating an empirical game). In the most typical approach, new strategies are added by calculating the best-response to some Nash equilibria of the current restricted game. If the generated best-response creates a deviation from the previous NE, it is added to the set of candidate strategies and the entire process is repeated. In the limit, we expect the calculated NE of the current game to converge to a NE of the full game. Thus, as we progressively add strategies to our candidate set, we begin to approximate Nash equilibria of the full game. Since generating a best-response to a profile is computationally very expensive (especially for large games), the EGTA/RL approach approximates a best-response using reinforcement learning.

The EGTA/RL approach can be summarized by the following broad stages:

1. Implement game simulator.

2. Select set of candidate strategies $\hat{S}$.

3. Estimate the empirical game.

4. Find a Nash equilibrium s^*.

5. Derive a new strategy L using RL, applied in a context where other agents play s^* and the learning agent attempts to best-respond.

6. Evaluate the learned strategy. If L provides a positive deviation from s^*, add L to $\hat{S}$, and extend the empirical game by continuing with stage 3. Otherwise, if learning has converged, and the RL model cannot be improved further, the process ends.

A figure representing the general EGTA/RL methodology is shown in Figure 1.

4 Set-up

4.1 Empirical Game

We devise the empirical game as a symmetric game of 3-player Texas Hold'em, with each game instance representing a play of 1000 hands. Each agent is given an infinite bankroll, and initialized to a starting position at the table (one of *Small Blind, Big Blind, and Dealer*), with each position rotating one seat at the end of every hand. Since a large amount of luck is involved in the game, we currently generate at least 25 sample game instances (or 25,000 hands) per profile, whose results are averaged when computing the

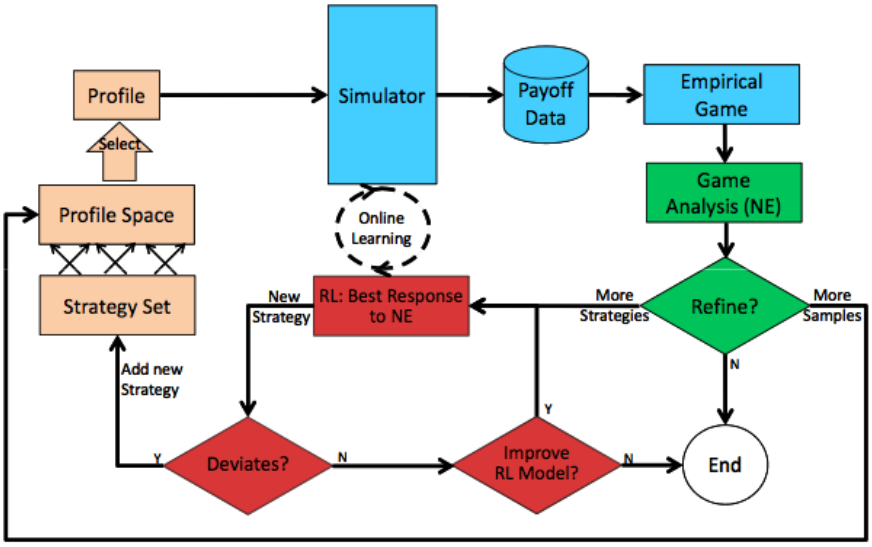


Figure 1: EGTA/RL Framework

estimated payoffs for each strategy. We note that each agent has its memory reset at the beginning of each sample game instance, thus any strategy that implements opponent modeling must be able to effectively do so within 1000 hands. We believe this limit is a good approximation to the number of played hands in a typical cash game.

We elect a symmetric representation of the game over a position-dependent, non-symmetric representation, (which is the case if each game instance is a single hand) because it both requires less profiles to sample and can easily encapsulate opponent modeling that forms over sequences of hands.

4.2 Initial strategies

We select an initial set of trivial strategies from which to generate our first restricted game. The initial strategies involve 3 agents, Callbot, Raisebot and Basicrulebot. Both the Callbot and Raisebot strategies are what they sound like: agents that always call and always raise, regardless of the current game state.

Basicrulebot uses a simple rule-based decision model that considers 2 unique features of its current state: $E[HS]$ and the *amount to call*. Basicrulebot will raise if it believes it has a strong hand, and will calculate implied odds when considering whether to fold. Furthermore, all of its actions are deterministic, and it makes no attempt to hide the value of its hand.

4.3 Learning Framework

The learning model we employ for generating an approximate best-response is online Q-learning.[?] We use Peng's Q(λ) algorithm, defining the learning model by a standard formulation of states, actions, and rewards.

4.3.1 State space

The standard game of three-player limit Hold'em has a state space of approximately 10^{24} states.[**?**] Obviously, it is not feasible to encode that many states in a learning model. To help ensure that we generate a new strategy with a non-negative payoff, we describe the state space with the same features that all 3 initial strategies use. Thus a learned response would, in theory, have to either be a positive best-response, or must exist in the support of the previous Nash equilibrium.

The learning agents use the features $E[HS]$, *amount to call*, and P_{pos} to describe the state space. Note that P_{pos} is the *positive hand potential*, or the probability that your current hand will improve in further rounds with new community cards. For example, a flush draw would have a high positive hand potential. P_{pos} is already implicitly used as a part of computing $E[HS]$, however we make it an explicit feature in our learning model to get a slightly higher fidelity abstraction of the actual state space. As both $E[HS]$ and P_{pos} are continuous variables, we use uniform bucketing with 10 buckets per feature. This gives us an upper bound of around $10 * 10 * 5 = 500$ differentiating states.

4.3.2 Actions and Rewards

Our initial action space consists of the three deterministic actions {call/check, raise, fold} except in states where *amount to call* is 0, in which case folding is dominated by checking.

We use both intermediate and terminal rewards in our learning model. As poker is a gambling game, we directly assign rewards to changes in the chip stack over states in a hand. Thus, raising in the current state will add $-(amtToCall + betSize)$ to the reward received in the next state. A terminal state occurs at the end of a hand, either after the final betting round has finished on the river, or all but one player has folded. Winning a hand will result in a positive terminal reward equal to the pot size won plus any existing reward appended from the previous state.

5 Experiments

We started our experiments by finding a Nash equilibria to the initial game containing our preliminary set of agents: {Callbot, Raisebot, BasicRuleBot }. Not surprisingly, both Callbot and Raisebot were dominated by BasicRuleBot, making (BasicRuleBot, BasicRuleBot, BasicRuleBot) a pure strategy Nash equilibrium of the restricted game.

5.1 09v10n

Our first approximated best-response, dubbed 09v10n, was generated after playing against 2 BasicRuleBots for 1,000,000 hands. In fact, since this was our first agent derived from reinforcement learning, we decided to learn a couple best-responses in parallel, in order to better explore the parameter space of the learning algorithms. Our initial parameter settings were set as $\gamma = 1$, $\delta = 1$, $\epsilon = .5$, $\lambda = 1$ and $\alpha = 1$. Both ϵ and α are decayed over time, with ϵ reducing linearly to 0 by the 1,000,000th hand, and α updating with $1/count^{0.5}$ after every observed learning state. Here *count* represents the number of times that the *learn* method has been called in the learning model.

We briefly explored different versions of ϵ and λ in our initial best-response testbed. We explored values of ϵ set at .5 and .25 (both reducing to 0 linearly over 1,000,000 hands) and values of λ set at 1 and 0.85. To compare these differing parameter values, we trained each agent with 20 batches of 50,000-hand matches, recording the score given at the end of each match. A graph summarizing the scores is shown in Figure **??**.

As shown in the figure, we can see that all of the agents performed pretty equally, especially by the 1,000,000th hand (where all the ϵ's eventually hit 0). ϵ did appear to converge a little faster early on, however our preliminary parameter experiments gave us a pretty strong confidence that our initial parameter settings would suffice.

After training, 09v10n was able to generate a winning rate of 0.788 sb/h against the BasicRuleBots, with each BasicRuleBot having a winning rate of -0.38 sb/h. Since an agent that always folds will receive -0.5 sb/h, we may generally want to view -0.5 sb/h as a realistic lower bound for an agent's winning rate, since if one were outmatched by a best-responder in a real game of Texas Hold 'em who was giving you a winning rate lower than -0.5 sb/h, you could simply adopt an AlwaysFold strategy.

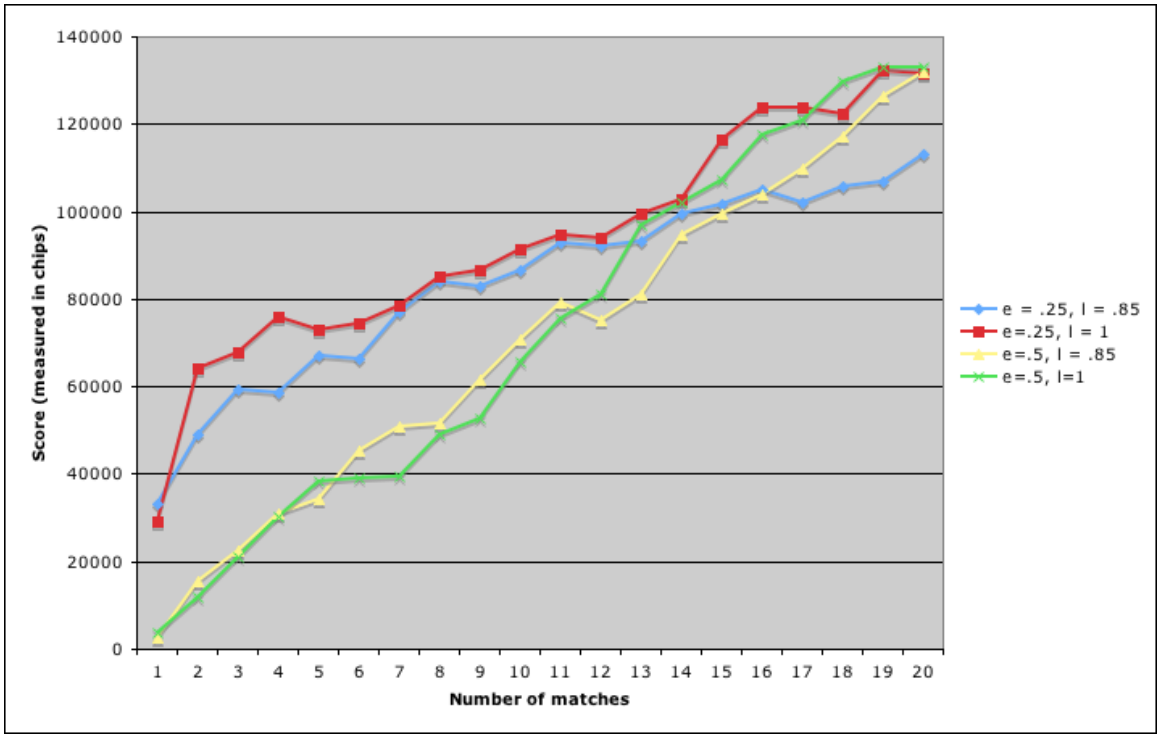


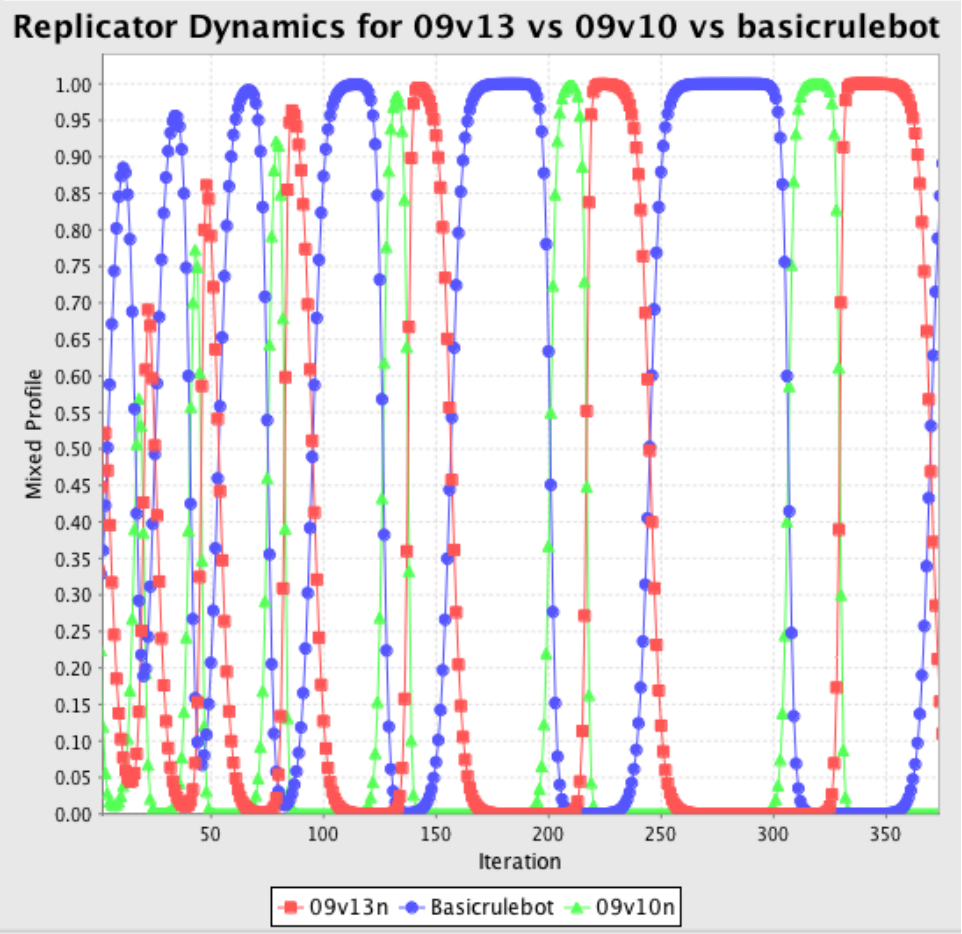
Figure 2: A collection of learning curves trained against 2 BasicRuleBots.

5.2 09v13n

With 09v10n clearly creating a positive deviation from the original Nash equilibrium, we created agent 09v13n as an approximated best-response to 2 09v10n agents. After 1,000,000 hands of training, we found that 09v13n was able to generate an incredible winning rate of 1.36 sb/h! Conversely, 09v10n had a winning rate of -0.6825 sb/h, which we recognize as a winning rate that is *worse* than the AlwaysFold strategy. While this could easily be conceived as great news, a strongly negative winning rate is largely indicative of a heavily exploitable strategy (which is the opposite thing we are looking for). Thus, while 09v10n was able to effectively best-respond to BasicRuleBot, its apparent exploitability leaves us to question the robustness of generated best-responses against other agents.

In fact, while 09v13n responds really well to a profile of 09v10n, it happens to perform poorly (i.e. have a negative winning rate) against BasicRuleBot. In an empirical game consisting of 09v13n, 09v10n and BasicRuleBot, *replicator dynamics*(which is the method we use to find Nash equilibrium in symmetric games) actually fails to converge to an equilibrium. Figure **??** shows a continuously oscillating mixed profile of the

three agents after 400 iterations of replicator dynamics.



5.3 09v14n

After 09v13n appeared to not be a very competitive strategy, we decided to produce an approximated best-response to a mixed strategy of 09v10n and BasicRuleBot. To implement this mixed strategy, 09v10n and BasicRuleBot were randomly selected (with equal probability) to play at the beginning of every hand. The goal was to see if a more robust best-response would be learned when playing against a mixed set of strategies. Luckily, 09v14n happened to be a more robust agent, with winning rates of 0.753 sb/h and 0.399 sb/h against 09v10n and BasicRuleBot respectively.

A new empirical game was created with 09v14n, with ϵ-Nash profiles shown in Figure **??**. As we can see, 09v14n is the newest pure strategy Nash equilibrium of our restricted game.

6 Opponent Modeling

Billings [1] described a method of opponent modeling that attempted to determine in detail what cards an opponent was holding. This was done by maintaining a weighted table of every possible hand an opponent might have, where the weights were the conditional probabilities than an opponent had played that particular hand to the current point in the game.

one	two	three	Epsilon
09v14n	09v14n	09v14n	0
09v14n	09v14n	Basicrulebot	219.538
09v14n	Basicrulebot	Basicrulebot	767.609
09v14n	09v14n	09v10n	961.92
09v14n	Basicrulebot	09v10n	971.961
09v14n	09v10n	09v10n	1,234.095
09v14n	09v14n	Callbot	1,261.25
09v14n	Basicrulebot	Callbot	1,393.781
09v14n	Callbot	Callbot	1,459.465
09v10n	09v10n	09v10n	1,499.279
Basicrulebot	Basicrulebot	Basicrulebot	1,566.597
09v14n	09v10n	Callbot	1,948.246
Basicrulebot	Basicrulebot	09v10n	2,072.017
Basicrulebot	Basicrulebot	Callbot	2,148.67
Basicrulebot	09v10n	09v10n	2,315.721
Basicrulebot	Callbot	Callbot	2,897.54
Callbot	Callbot	Callbot	3,066.84
09v10n	Callbot	Callbot	3,074.1
Basicrulebot	09v10n	Callbot	3,420.08
09v10n	09v10n	Callbot	4,587.469

Figure 3: The top ϵ-Nash profiles of a game consisting of 09v10n, 09v14n, BasicRuleBot, and Callbot

It is of note that the University of Alberta's most successful agents, which are among the best in the world, have not used this method. In fact recent conversations with Michael Bowling, the leader of Alberta's Poker Research Group have indicated that, despite having spent a great deal of time on it, their group has made little progress on opponent modeling of any kind. We have not been dissuaded by their lack of success.

Our approach has dispensed with trying to determine what specific cards a player is holding and to concentrate on modeling betting behavior, depending upon the round, by looking at only HS or $E[HS^2]$. To do this we construct a kernel density estimate (KDE) for each possible amount that a player can bet for each round in the game. This gives a total of 20 KDEs, one for each of the 5 possible amounts bet per round, however the KDEs differ between rounds. For the *preflop* and *river* they model the density of the HS, while for the *flop* and *turn* the KDEs model the density of the $E[HS^2]$.

The KDEs are constructed during the course of a match by observing a player's cards during a particular showdown and then retroactively calculating his HS and $E[HS^2]$ for each of the prior rounds. Then based upon the amount he bet in each round we select the corresponding KDE and add a point to it. In this way the model of a player's betting behavior is incrementally improved throughout the course of a match.

The process just described is how the opponent model would work during an actual game, however for this project, rather than play actual games, we opted to simulate games using the logs of the 2009 Computer Poker Competition. These log files, which record the full betting and card histories of over 5300 matches played by the 9 entrants of the competition, are the only source of information we have about the behavior of the top poker agents. Therefore, because we want to evaluate our opponent modeling methodology on the best possible players, simulating games based upon the content of the logs made better sense than playing live games against mediocre agents. In addition, because of the number of logs files, it is possible run a large variety of repeatable experiments with reasonable computational overhead.

To evaluate the performance of the opponent model we inspected a player's hole cards and then, depending upon the round, calculated his HS or $E[HS^2]$. We then took the resulting metric and consulted our KDEs to determine a bound on the player's betting behavior. This was done by consulting each of the particular round's five KDEs and determining the probability that the player would bet each amount given our previously calculated metric. We then normalized the five resulting probabilities and used those to calculate the player's expected bet size. Fractions were rounded. This calculation was always done *prior* to adding the point to the KDE.

Next, we calculated how well the expected bet size corresponded to the player's actual betting behavior by determining the percentage of hands for which we were able to put an upper bound on the player's bet. In addition we also measured the error, in average number of bets per hand, that was incurred when we both successfully and unsuccessfully bounded the bet size. Finally, in the cases that we successfully determined

an upper bound we calculated the average per hand error for all of the hands in which the player's last action was a call or a fold. This final metric, called *call and fold error per bounded hand* (CFEPBH), was calculated because we wanted to understand the quality of our estimates given that the player did not increase his bet. In contrast a player whose last action in a round was raise may have been willing to bet even more given the opportunity. Overall a high percentage of correctly bounded hands combined with a low CFEPBH would indicate that our modeling efforts were at least relatively successful.

The first application of the opponent model was to test its feasibility by allowing the addition of data points to the relevant KDEs based upon a player's cards regardless of whether the cards would have been shown during actual game play. In addition, predictions were not considered for our statistics until after the 500th hand of any match had been simulated. Our contention was that if we couldn't be reasonably accurate in this relaxed scenario then the prospects of modeling a player during the course of a game were bleak. On the other hand, if the opponent model performed well in the relaxed game, then the results from this relaxed simulation would provide us with a reasonable upper bound on the opponent model's performance when the rules were enforced and predictions proceeded from the very first hand.

Table 6 shows the values of the metrics describe above for each of the top three agents in the 2009 competition averaged across 50 different matches along with their standard deviations.

	PB	**EPUH**	**EPBH**	**CFEPBH**
Hyperborean-Eqm				
Preflop	85.5 ± 3.4	1.09 ± 0.04	0.80 ± 0.07	0.55 ± 0.07
Flop	94.3 ± 2.3	1.29 ± 0.16	1.06 ± 0.14	0.61 ± 0.12
Turn	94.5 ± 2.5	1.15 ± 0.14	0.74 ± 0.14	0.48 ± 0.12
River	90.4 ± 4.2	1.14 ± 0.15	0.79 ± 0.13	0.43 ± 0.12
Total	89.8 ± 2.0	1.12 ± 0.05	0.85 ± 0.06	0.53 ± 0.07
Hyperborean-BR				
Preflop	83.5 ± 2.9	1.07 ± 0.04	0.84 ± 0.05	0.61 ± 0.05
Flop	93.0 ± 2.1	1.21 ± 0.12	0.91 ± 0.12	0.53 ± 0.09
Turn	93.7 ± 2.5	1.15 ± 0.14	0.74 ± 0.18	0.46 ± 0.12
River	91.0 ± 3.6	1.09 ± 0.08	0.83 ± 0.19	0.46 ± 0.13
Total	88.6 ± 1.7	1.10 ± 0.03	0.84 ± 0.07	0.54 ± 0.19
dpp				
Preflop	83.3 ± 4.4	1.04 ± 0.03	0.84 ± 0.10	0.78 ± 0.10
Flop	91.4 ± 2.2	1.17 ± 0.11	0.83 ± 0.09	0.43 ± 0.11
Turn	91.6 ± 2.9	1.15 ± 0.12	0.80 ± 0.11	0.34 ± 0.11
River	87.1 ± 4.2	1.17 ± 0.12	0.92 ± 0.09	0.30 ± 0.10
Total	87.6 ± 2.5	1.09 ± 0.04	0.84 ± 0.06	0.52 ± 0.07

Table 1: The Percentage of Hands Successfully Bounded (PB), Error Per Unbounded Hand (EPUH), Error Per Bounded Hand (EPBH) and the Call and Fold Error Per Bounded Hand (CFEPBH) for the top three finishers in the 2009 Poker Competition using the relaxed simulation. Note that for the rows marked *Total* the statistics are heavily skewed due to the fact that the number of samples for the *preflop* are much larger the number of samples for any of the other rows. This caveat also applies to Table **??**.

A few things are worth noting. First, our ability to bound the betting behavior of any of the agents during the *preflop* is considerably worse than for any other round. Fortunately this might be improved by using $E[HS^2]$ instead of HS as the metric for our *preflop* KDEs. Next, when we underestimate an opponent's bet we are off on average by over 1 bet, however we do not underestimate that often. Finally, the CFEPBH shows that we are on average only overestimating an opponent's bet by 0.5 bets per round. Overall, in the relaxed case the opponent model is doing a decent job of bounding the betting behavior. It is definitely worth considering what happens when the relaxations are removed and we only allow our opponent model to use the information that would be revealed during a real game.

The results for the unrelaxed simulations, which used the exact set of log files that the relaxed simulations

	PB	EPUH	EPBH	CFEPBH
Hyperborean-Eqm				
Preflop	91.4 ± 2.1	1.10 ± 0.05	0.95 ± 0.18	0.70 ± 0.14
Flop	90.4 ± 3.3	1.37 ± 0.13	0.86 ± 0.14	0.51 ± 0.11
Turn	84.8 ± 6.0	1.15 ± 0.09	0.59 ± 0.12	0.35 ± 0.12
River	85.6 ± 3.5	1.11 ± 0.08	0.70 ± 0.13	0.36 ± 0.11
Total	89.1 ± 1.9	1.16 ± 0.04	0.84 ± 0.10	0.55 ± 0.10
Hyperborean-BR				
Preflop	90.9 ± 2.6	1.09 ± 0.05	1.04 ± 0.17	0.78 ± 0.15
Flop	88.8 ± 2.7	1.24 ± 0.08	0.76 ± 0.10	0.45 ± 0.08
Turn	89.1 ± 3.9	1.17 ± 0.08	0.64 ± 0.13	0.39 ± 0.09
River	86.2 ± 4.2	1.11 ± 0.07	0.73 ± 0.18	0.39 ± 0.12
Total	89.4 ± 1.5	1.14 ± 0.04	0.86 ± 0.10	0.58 ± 0.10
dpp				
Preflop	92.0 ± 3.4	1.10 ± 0.09	1.00 ± 0.15	0.93 ± 0.14
Flop	90.3 ± 1.9	1.21 ± 0.09	0.80 ± 0.07	0.42 ± 0.09
Turn	90.2 ± 2.4	1.17 ± 0.09	0.76 ± 0.08	0.33 ± 0.08
River	85.3 ± 4.5	1.15 ± 0.09	0.92 ± 0.08	0.27 ± 0.09
Total	90.1 ± 1.7	1.14 ± 0.05	0.89 ± 0.07	0.58 ± 0.07

Table 2: This table is identical to Table 6 except that the simulations used to generate the data were unrelaxed.

did, are shown in Table **??**.

One surprising result is that our ability to bound *preflop* betting behavior is improved, although our error in doing so is unfortunately high. But again using $E[HS^2]$ might allow for some improvement with this. Another interesting result is that, other than the *preflop*, error per hand measures are all lower for the unrelaxed simulations. Unfortunately the reasons for this are unclear. On the other hand, the percentage of hands successfully bounded is lower and the error for unbounded hands is generally higher.

7 Conclusions and Future Directions

While we were only able to produce a few iterations, our EGTA/RL approach was able to expand upon an initial set of trivial agents. There were a couple issues questioning the robustness of our learned best-response strategies, however learning against a mixed strategy appeared to be very helpful in generating a more robust strategy.

It would have been helpful to have had access to previous poker strategies (for example, from the Benchmark server of the 2009 Computer Poker Competition), as it is very probable that some of the difficulties involved in producing a robust best-response very well could have been caused by a weak initial set of trivial agents, all of whom are noticeably exploitable. Future directions for our poker agents will involve generating a higher fidelity state space of the poker game to use in our learning models, as well as potentially adding opponent modeling behavior to agents.

Overall the opponent model's performance is okay, but there are several things that might be done to improve it. First, using $E[HS^2]$ instead of just the HS on the preflop would probably improve our rather poor preflop statistics. Next, moving to a multivariate density where we consider other metrics like implied odds could also be beneficial.

It would also be good to more fully understand the conditions where we fail to bound a player's betting behavior and to understand more precisely the source of our errors. For example, are there certain types of hands, perhaps ones where the opponent is bluffing, where the expected value calculation just does not

make sense? In fact the expected value calculation may not be the best way to evaluate the opponent model at all.

Whatever improvements we may make, there are is one possible use of our opponent model that we would like to explore. Basically, given the competition logs, we should be able to build a fairly accurate profile of the various agents' behaviors over many matches. From there it should be an easy exercise to build our own agents that use the opponent model as the means for making betting decisions. That is, using our opponent model we might be able to make a lossy copycat agent for every agent that entered the tournament.

If these copycat agents were even semi-competent they could prove valuable for training our RL agent. On the other hand, if the copycat agents are generally poor poker players then this line of research may be at an end. In fact, creating an agent based upon an opponent model and showing that it can play a good game of poker is an essential step to take before finally using the opponent model in our RL agent.

References

[1] Billings, Darse, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. "The Challenge of Poker." Artificial Intelligence Journal, vol 134(1-2). 2002.

[2] Billings, Darse, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. "Approximating Game-Theoretic Optimal Strategies for Full-scale Poker." Proceedings of the 2003 International Joint Conference on Artificial Intelligence. 2003.

[3] Gilpin, A, and T. Sandholm. "Lossless abstraction of imperfect information games." Journal of the ACM, 54 (5), October. 2007.

[4] Gilpin, A, and T. Sandholm. "Expectation-Based Versus Potential-Aware Automated Abstraction in Imperfect Information Games: An Experimental Comparison Using Poker." Proceedings of the 2008 AAAI Conference on Artificial Intelligence. 2008.

[5] Schvartzman, LJ, and MP Wellman. "Stronger CDA Strategies through Empirical Game-Theoretic Analysis and Reinforcement Learning." Eighth International Joint Conference on Autonomous Agents and Multiagent Systems, pages 249-256, May 2009.

[6] X. Chen and X. Deng. "Settling the complexity of 2-player Nash equilibrium". In Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS), 2006.

[7] Wikipedia: The Free Encyclopedia. Texas Hold'em. 13 December 2009. `http://en.wikipedia.org/wiki/Texas_hold_'em`.

[8] M.B. Johanson. "Robust strategies and counter-strategies: Building a champion level computer poker player." Master's thesis, Computing Science Department, University of Alberta, Edmonton, 2007.

[9] Kevin Leyton-Brown and Yoav Shoham. "Essential of Game Theory: A Concise, Multidisciplinary Introduction". Morgan and Claypool, 2008.

[10] Richard S. Sutton and Andrew G. Barto. "Reinforcement Learning". Cambridge: The MIT Press, 1998.

[11] Nicholas Abou Risk. "Using Counterfactual Regret Minimization to Create a Competitive Multiplayer Poker Agent." Master's thesis, Computing Science Department, University of Alberta, Edmonton, 2009.

[12] M. P. Wellman. "Methods for empirical game-theoretic analysis (extended abstract)."" In Twenty-First National Conference on Artificial Intelligence, pages 1552-1555, Boston, 2006.