Particle Filtering for Dynamic Agent Modelling in Simplified Poker

Nolan Bard and **Michael Bowling**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{nolan,bowling}@cs.ualberta.ca

Abstract

Agent modelling is a challenging problem in many modern artificial intelligence applications. The agent modelling task is especially difficult when handling stochastic choices, deliberately hidden information, dynamic agents, and the need for fast learning. State estimation techniques, such as Kalman filtering and particle filtering, have addressed many of these challenges, but have received little attention in the agent modelling literature. This paper looks at the use of particle filtering for modelling a dynamic opponent in Kuhn poker, a simplified version of Texas Hold'em poker. We demonstrate effective modelling both against static opponents as well as dynamic opponents, when the dynamics are known. We then examine an application of Rao-Blackwellized particle filtering for doing dual estimation, inferring both the opponent's state as well as a model of its dynamics. Finally, we examine the robustness of the approach to incorrect beliefs about the opponent and compare it to previous work on opponent modelling in Kuhn poker.

Introduction

Agent modelling is the problem of building a predictive model of another agent's future decisions from, possibly incomplete, observations of past behavior. The rise of applications requiring interaction between independent agents — human, robotic, and software — has made coping with the presence of other decision makers a key challenge for artificial intelligence. Agent modelling is one approach to this challenge. With accurate predictive models of the other agents, an effective response can be planned and executed. Applications as diverse as assistive technologies, autonomous driving, electronic commerce, and interactive entertainment all require or would benefit significantly from accurate models of other agents, artificial or human.

There are many factors that complicate the agent modelling task. Models often need to be constructed from very few observations. The environment may involve stochastic events or hidden information only known to the agent being modelled. The agent itself may, in fact, make stochastic choices. And finally, the agent may be dynamic, changing their behavior throughout the interaction. For example, a long-term customer's product preferences can drift over time, a web visitor's goals can change while browsing a site, or a game opponent may switch strategies during play.

In this work, we approach the agent modelling problem as a specific instance of the general problem of *state estimation*. State estimation involves tracking a stochastic process's hidden state variables by observing noisy functions of these variables. Such techniques represent uncertainty in the hidden state as a probability distribution and use a recursive Bayes' formula to update the belief with each new observation. Although these techniques require a complete probabilistic model of the process dynamics and observations, techniques for *dual estimation* have relaxed this requirement by inferring these models from data. State estimation seems to be well-suited to the agent modelling task, and yet it has received little attention in the agent modelling literature.

Poker is a challenging testbed for AI research that embodies all of the agent modelling challenges. The environment and often the players are stochastic. Each player has their own private information which may never be revealed. In addition, modelling is expected to play a key role in the eventual development of world champion poker-playing programs.[1] Dynamic behavior is also an integral part of expert poker. Exploiting one's "table image" by "switching gears" is a common tactic, which involves drastically changing play style. This makes poker an ideal domain for examining the problem of dynamic agent modelling. In particular, we will focus on a simplified version called Kuhn poker.

This paper examines the use of the state estimation technique, particle filtering, along with an extension to dual estimation, for the problem of agent modelling in poker. We begin with background on agent modelling and state estimation. Next, we introduce Kuhn poker, and describe our application of particle filtering to this domain. We then show results of our approach on a broad range of scenarios, including when the opponent's dynamics are known, inferred,

[1] Many of the competitive poker playing programs today employ a game-theoretic approach without any opponent modelling (Billings *et al.* 2003; Gilpin & Sandholm 2006). These approaches, though, by failing to identify and exploit their opponents' weaknesses are unlikely to best the world's top human players. This was also the observation of a world class player after evaluating one of these game theoretic programs, "You have a very strong program. Once you add opponent modelling to it, it will kill everyone." (Billings *et al.* 2004)

and even wrong. These results illustrate both the effectiveness and robustness of the technique. Finally, we conclude.

Background

This work brings ideas from state estimation to the problem of agent modelling. We briefly review past work in agent modelling, before describing the necessary foundational concepts from state estimation.

Agent Modelling

Agent modelling has been explored in the literature under a number of different guises. Early work, which continues today, focused on plan recognition (e.g., Kautz 1991) as the inverse of the traditional planning problem: recovering a goal concept from a sequence of actions in a symbolic planning language. If the agent's goals can be deduced, one can then predict the agent's actions in novel situations. This inversion of planning has also been attempted for stochastic domains as the problem of inverse reinforcement learning (Ng & Russell 2000), where a reward function is identified from samples of the agent's policy. These techniques, though, generally require complete knowledge of the rest of the system, restricting them to situations of modelling a single agent making deterministic choices in a fully observable environment.

Behavior classification (Han & Veloso 1999) and policy recognition (Bui, Venkatesh, & West 2002) try to identify behavior rather than goals. They were proposed for partially observable, stochastic domains and take a more probabilistic approach that can even account for an agent making stochastic decisions. However, they only recognize or classify observed behavior into a small discrete set of target behaviors that must be specified in advance. Also, these approaches (along with plan recognition and inverse reinforcement learning) assume the agent's behavior is static, unchanging throughout the interaction.

Agent modelling has also been investigated in the realm of game theory, particularly in the context of small strategic form games such as prisoner's dilemma. For example, Carmel and Markovitch (1996) proposed a technique for modelling an "opponent" as a deterministic finite automaton, which can then be used for planning an optimal response. Although the approach can learn models of dynamic agents, it is not obvious how it can be extended to stochastic choices, private information, or larger games.

Previous work on agent modelling in Texas Hold'em poker has focused on techniques for modelling static agents. Simple frequentist opponent modelling systems have been used to modify the distribution over the opponent's hidden cards based on their past behaviour (Billings *et al.* 1998). Artificial neural networks were used to learn opponent models offline from previous game data. The models were then used for choosing salient features to augment previous frequentist modelling (Davidson *et al.* 2000). Vexbot, the foremost opponent modelling poker program, incorporates frequentist statistics into an imperfect information game-tree search (Billings *et al.* 2004). These techniques, however, require considerable data to be effective. Recent work by Hoehn and colleagues (Hoehn *et al.* 2005) focused on short-term modelling in Kuhn poker. We will discuss this work in more detail below, along with a comparison of results. All of this previous work in poker opponent modeling, however, is focused on modelling static opponents, where as our work explicitly models dynamic behavior.

State estimation has been wildly successful in a broad range of applications. Although seemingly well-suited to the challenges of agent modelling, it has received very little attention in this area. Bererton (Bererton 2004) proposed the use of particle filtering in commercial computer games, but this was to imbue computer players with more realistic beliefs about enemy positions. We propose to use state estimation techniques to infer an agent's subjective state, i.e., its behavior, rather than any objective quantities.

State Estimation

State estimation is the problem of determining the current state of a system given a sequence of observations. This is done by representing the uncertainty in the current state as a probability distribution and using Bayes' rule to update the belief after every observation.

Formally, let x_t be the state vector, and z_t be the observation vector at time t. Define $x_{1:t}$ to be the state sequence $x_1, \ldots, x_t$ and similarly for the observation sequence $z_{1:t}$. Then the state estimation problem is concerned with estimating $x_T | z_{1:T}$, or rather the complete distribution $\Pr(x_T | z_{1:T})$. By applying Bayes' rule, simple arithmetic, and assuming that x_t is a sufficient statistic for the events up to time t (i.e., the Markov assumption), we arrive at the standard recursive Bayesian filtering equation,

$$\begin{aligned}\Pr(x_t|z_{1:t}) &= \eta \Pr(z_t|x_t) \\ &\quad \int \Pr(x_t|x_{t-1}) \Pr(x_{t-1}|z_{1:(t-1)}) dx_{t-1},\end{aligned}$$

where η is a normalization constant. Given our previous belief $\Pr(x_{t-1}|z_{1:(t-1)})$, we can use this equation to find our new belief after the latest observation. The equation requires an *observation model* $\Pr(z_t|x_t)$ and a *motion model*[2] $\Pr(x_t|x_{t-1})$. Lastly, for a practical implementation the form of the belief distribution $\Pr(x_t|z_{1:t})$ needs to allow the integral in the Bayesian filtering equation to be computed easily. For example, a Gaussian form results in a Kalman filter and its assorted variants. A Monte Carlo approximation results in a particle filter, which is the approach taken in this paper.

Particle Filters. Particle filters are a Monte Carlo approach to Bayesian filtering. Particle filters approximate the probability distribution over the state using a set of samples called *particles*. Each particle is a state vector, which we denote as $x_t^{(i)}$. Particle filters are flexible and powerful. They can handle non-linear dynamics while representing arbitrary belief distributions over the state variables. The accuracy and computational cost of a particle filter scales with the number of particles used.

[2] Often the motion model will include other terms such as a control input signal u_t. Since our application does not involve any additional signals we exclude them from our presentation.

The particle filter update is based on importance sampling. Given a particle $x_{t-1}^{(i)}$ approximately sampled from $\Pr(x_{t-1}|z_{1:(t-1)})$ we want a new particle $x_t^{(i)}$ approximately sampled from $\Pr(x_t|z_{1:t})$. We do this by sampling $\tilde{x}_t^{(i)}$ from the candidate distribution $\Pr(x_t|z_{1:(t-1)})$ and weighting it by the importance sampling correction $\Pr(z_t|\tilde{x}_t^{(i)})$. Sampling from the candidate distribution involves sampling from the motion model $\Pr(x_t|x_{t-1}^{(i)})$. The weighting comes from the observation model. In order to turn the samples from the candidate distribution into samples from our desired distribution we then select n particles, with replacement, randomly in proportion to their weights.

Dual Estimation. Here we consider a simple case of dual estimation where the system's dynamics are parameterized by some unknown value θ, which we need to simultaneously infer along with the system state (Doucet, de Freitas, & Gordon 2000; Storvik 2002). We can do this through an application of Rao-Blackwellized particle filters (RBPFs), a hybrid state estimation technique that allows a portion of the state variables to be modelled in a Monte Carlo fashion while others are modelled in a parametric form. For each particle, we additionally store a sufficient statistic $s_t^{(i)}$ for computing $\Pr(\theta|x_{1:t}^{(i)})$. Sampling from the candidate distribution now involves sampling $\tilde{\theta}$ from $\Pr(\theta|s_{t-1}^{(i)})$ and then sampling $\tilde{x}_t^{(i)}$ from $\Pr(x_t|x_{t-1}^{(i)}, \tilde{\theta})$. The sufficient statistic for each candidate particle is updated for the new transition $\tilde{s}_t^{(i)} = \textsc{Update}(s_{t-1}^{(i)}, x_{t-1}^i \rightarrow \tilde{x}_t^i)$. The weight from the observation model and resampling is performed in the usual fashion. As long as we choose $\Pr(\theta)$ carefully with an appropriate sufficient statistic this adds very little additional running time to the basic particle filtering algorithm.

Kuhn Poker

Kuhn poker is a tiny, toy variant of poker for which a complete game theoretic analysis exists (Kuhn 1950). The game involves two players; two actions, *bet* and *pass*; and a three card deck, containing a Jack (J), Queen (Q) and King (K). Each player is dealt one card privately. The first player may then either bet or pass. If the first player bets the second player may either bet, causing a *showdown*, or pass, to *fold*. If the first player passes, the second player can also pass, causing a showdown, or bet, forcing the first player to make a final decision of either bet, for a showdown, or pass, for a fold. In the case of a fold, the non-folding player wins one dollar from the folding player. In the case of a showdown, the player with the higher card (King is high, Jack is low) wins one dollar if neither bet, or two dollars if both players bet.

With its introduction, Kuhn also presented a complete game theoretic analysis of the game. Although there are 64 different pure strategies for each player, many of these are dominated, i.e., another strategy has a higher expected value against every possible strategy of the opponent. After eliminating dominated strategies, the undominated strategy space of player one can be parameterized by three parameters (α, β, γ), and player two by two parameters (η, ξ). These parameters are all in the range $[0, 1]$ and specify the probability of betting in certain information sets. For example, η is the probability the second player bets when facing a bet while holding the Queen, and ξ is the probability the second player bets after a pass when holding the Jack. The game has a whole continuum of Nash equilibria, which can be written in this parameterization as , $\alpha = \gamma/3$, $\beta = (1 + \gamma)/3$, and $\eta = \xi = 1/3$. The value of the equilibrium is $-1/18$ dollars per game. In other words, if either player plays an equilibrium strategy then player one will lose 5.5 cents per game on average.

Although the equilibrium strategy guarantees a minimum payoff regardless of the opponent's strategy, greater payoffs may still be possible. For example, suppose player two is choosing randomly among undominated actions (i.e., $\eta = \xi = 0.5$). If player one persists with an equilibrium strategy, the loss of $-1/18$ per game is unchanged. If player one instead responds by passing in the first round and betting with a King or Queen when bet to (i.e., the best response), it is no longer a losing game, with the expected payoff being zero. Other deviations from the equilibrium strategy can be exploited by even more. It is clear that an accurate model of the opponent's strategy can be used to great advantage.

Kuhn poker is an ideal domain for our investigation. It has most of the strategic properties found in the real world variations of poker, yet a game theoretic analysis of the game is tractable and exists. For instance, Kuhn preserves the ability to trap (i.e., acting as though your hand is weaker than it truly is) and bluff (i.e., acting as though your hand is stronger that it truly is). In Kuhn we bluff by betting with a Jack and trap by passing with a King. In addition, players' strategies have a natural parameterization, and best-response is a straightforward computation. Since these features are all active research directions in full versions of poker, Kuhn poker offers a clean domain for evaluating ideas. It has also already been a testbed for modelling research. Hoehn and colleagues (Hoehn *et al.* 2005) examined two modelling techniques for exploitation of *static* opponents. We will return to their results later.

Application to Kuhn Poker

We now describe how we adapt the particle filtering technique to Kuhn poker. To simplify things, we will restrict ourselves to the situation of player one modelling player two. To use particle filtering, we need to define five components: the state variables, the observations, an observation model, a motion model, and an initial belief.

For the state variables, one natural choice is simply Kuhn's own parameterization of the strategy space. This choice involves two assumptions (i) our opponent will not play outside the parameterization (i.e., does not play dominated strategies), and (ii) our opponent's future strategies are conditionally independent of past strategies given their current strategy (i.e., the Markov property). For our situation of modelling player two, this means the state vector x_t is two-dimensional with the values η_t and ξ_t.

As the particle filter is being updated after each hand, the observation z_t naturally consists of any known cards and the betting sequence for that hand. The observation model $\Pr(z_t|\eta_t, \xi_t)$ comes from the definition of the game itself and is a straightforward computation. With folded hands we are still able to compute the likelihood by marginalizing over the unknown cards, which with our small deck is a simple operation. An encoding of these probabilities can be easily worked out by hand due to the small size of Kuhn and its simple strategy parameterization.

The motion model must encode our belief about how our opponents will change their strategies over time. In this paper we will explore two naive types of motion models. The first model assumes that players will change with probability ρ to a random strategy after every hand. With probability $(1 - \rho)$ they continue using their previous strategy. We call this a *switching* model. The second model assumes that players' strategies drift after each hand. The player generates their next strategy by sampling from a spherical Gaussian distribution with a mean of the current strategy and variance σ^2, restricting it to the unit square, and re-normalizing appropriately. We also refer to a combined motion model which involves both the ρ and σ parameters. In this model, the player switches to a uniform random strategy at the end of the hand with probability ρ and with probability $(1 - \rho)$ it drifts with variance σ^2.

Finally, we use a uniform random distribution over the state space as our initial belief $\Pr(x_0)$.

Dual Estimation. As we have chosen motion models that are parameterized, we can employ an RBPF to infer the motion model parameter, either ρ or σ. The only detail is specifying a prior on ρ and σ such that the posterior can be summarized with a sufficient statistic. For ρ we can model the prior and posterior as a beta distribution $\textsc{Beta}(\alpha, \beta)$ where α and β are the sufficient statistics. These are updated easily on observing a transition: if $x_{t-1} = x_t$ increment β, otherwise increment α. For σ^2 we model the prior and posterior as an inverse-gamma distribution $\textsc{Inv-Gamma}(v, w)$ where v and w are the sufficient statistics. Because we use a spherical Guassian, transitions are treated as one observation of σ^2 per dimension. On observing a transition the update adds $||x_t - x_{t-1}||^2/2$ to w and $d/2$ to v, where d is the number of dimensions in the state vector (two, in our case).

Finally, we need to choose a prior belief for ρ or σ, depending on the motion model in use. For the experiments in the next section we fairly arbitrarily chose $\Pr(\rho) = \textsc{Beta}(1, 30)$ and $\Pr(\sigma^2) = \textsc{Inv-Gamma}(0.6, 0.00005)$.

Using Our Model. One final consideration is what to do with our belief about the opponent's strategy. The correct Bayesian approach is to select the action that maximizes the sum of all future expected utility given our belief. This is, in general, an intractable computation. Instead, we opt for the simpler greedy response to maximize expected utility for the current hand given our belief. Because of the linearity of the expected value in our chosen state variables the best-response to our particle posterior is just the best-response to the mean of the particles, which can be computed quickly. We will simply play this best-response to our posterior distribution on every hand.

Table 1: Total expected winnings of Hoehn's parameter estimation algorithm compared to our particle filter with a stationary motion model over 200 hands.

Opponent	Particle Filter	Parameter Estimation
$O_1 = (0.8, 0.29)$	4.3	-9.0
$O_2 = (0.75, 0.8)$	18.7	9.0
$O_3 = (0.67, 0.4)$	-2.7	-9.0
$O_4 = (0.17, 0.2)$	2.5	3.0
$O_5 = (0.25, 0.17)$	-1.3	-2.5
$O_6 = (0.25, 0.67)$	10.6	16.0

Results

We now evaluate our various modelling approaches against a variety of opponents, both static and dynamic, and when our prior beliefs are both correct and incorrect. This will help us to understand both the effectiveness and robustness of the approach. We begin by briefly examining static opponents and then move on to dynamic opponents.

Static Opponents

Hoehn and colleagues' previous work on opponent modelling in Kuhn poker focused on static opponents (Hoehn *et al.* 2005; Hoehn 2006). We begin by replicating one of their experiments with our particle filtering approach. They examined two modelling approaches: one using explicit parameter estimation with similar Bayesian roots as our approach and an implicit modelling technique similar to an "experts" algorithm. For 200 hand matches, they found the parameter estimation approach with a fixed 50 hand exploration period worked the best. We played a particle filter modeller using a stationary motion model (since all of the opponents were stationary) against their six static opponents. Table 1 shows the results. The particle filter approach is competitive (performing better against some opponents and worse against others) with Hoehn's parameter estimation approach. However, our approach was specifically designed to handle dynamic agents, so we now move on to those opponents.

Dynamic Opponents

We compared various incarnations of the particle filter against several simple dynamic agents. The simple agents employ one of our two motion strategies described above: either switching randomly after each hand or moving according to the truncated spherical Gaussian drift after each hand. The opponents use the same parameterization as the model with ρ specifying the switching probability for the switching opponents and σ the drift standard deviation for the drift opponents. Nine specific opponents (4 switching, 4 drifting, and 1 stationary) were chosen, with the specific values of ρ and σ given in Table 2. These values were chosen to cover opponents that moved both slowly and quickly.

We played various incarnations of particle filtering against each of our nine opponents for 1000 hand matches. These matches were repeated 5000 times for statistical confidence. Recall that since these experiments involve player one modelling player two, this is actually a "losing" situation for the

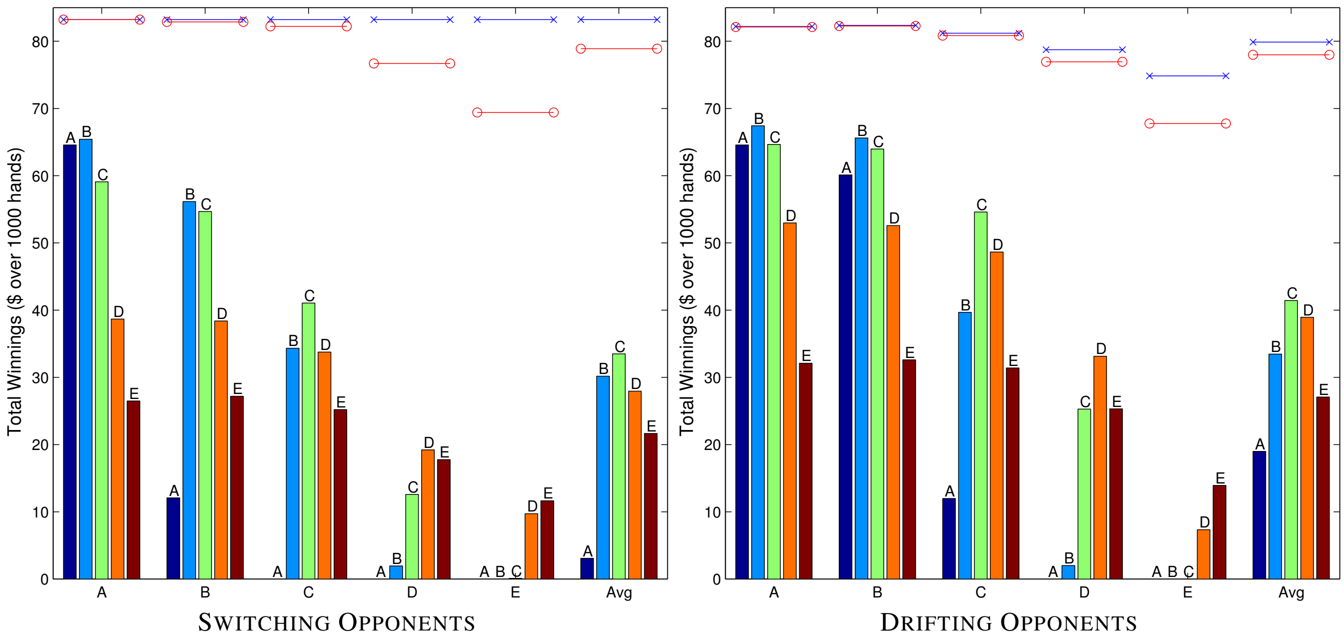

Figure 1: Total winnings of vanilla particle filter modelling against different switching opponents (left) and drifting opponents (right). Each bar represents a different setting of ρ (left) or σ (right) in the filter's motion model.

Table 2: Opponent Legend

	Motion Model	
Opponent	Switch (ρ)	Drift (σ)
A	0.0	0.0
B	0.002	0.005
C	0.01	0.02
D	0.05	0.05
E	0.1	0.1

modelling player. If the modelling player instead played any equilibrium strategy it would lose over $55 on average over 1000 hands. A less naive alternative would be to exploit the fact that all of the described opponents are effectively random. Without considering the knowledge of past play, they will make uniformly random decisions between any undominated choices on every round. The best-response to such random play is an alternative static strategy that gains and loses nothing on average against any of our opponents over 1000 hands. This makes zero a good baseline comparison for which effective opponent modelling should rise above.

To provide further context to our results, we display two exploitability lines for each opponent. The top line in each graph is the expected exploitability if we knew our opponent's strategy on every hand. This is the upper bound on any agent's expected performance. The bottom line is the expected exploitability if we knew our opponent's strategy on their previous hand and their motion model. This baseline is a more practical goal than the first exploitability line since it is more sensitive to how quickly the particular opponent's strategy is changing. Both lines are, however, only loose upper bounds on possible performance, as it takes many hands of Kuhn poker to estimate even a static opponent's strategy parameters.

Figure 1 gives a summary of the results of applying various incarnations of particle filtering to these opponents. The left graph shows the results for the switching opponents and the right graph shows the results for the drift opponents. In both graphs, the x-axis corresponds to the different opponents from Table 2. For each opponent the set of bars show the average total winnings of particle filtering when using the correct form of the motion model, but varying the parameter in that model (i.e., ρ or σ). The parameters took on the same set of values as the opponents and so each bar is labeled with the corresponding opponent's identifier from Table 2. We dissect this figure further below.

Known Motion Model. Consider the case when the opponent's motion model is known and correct. Specifically, we know whether the opponent is a switching or drifting player as well as the exact probability of switching or the average amount of drift. This corresponds to the bars of Figure 1 that match the opponent being played on the x-axis. Although the winnings vary depending on the opponent, for all nine opponents the correct model outperformed the baseline static strategy which nets zero.

Realistically, knowing the exact parameterization of another agent's motion is unlikely. It is interesting to consider how the approach performs if its model is wrong. This case corresponds to the remaining bars in Figure 1. Naturally, performance is higher when the opponent's motion parameter is close to the model's parameter. The fall off for an incorrect model, though, can be quite drastic, sometimes dropping lower than our simple static baseline performance. For

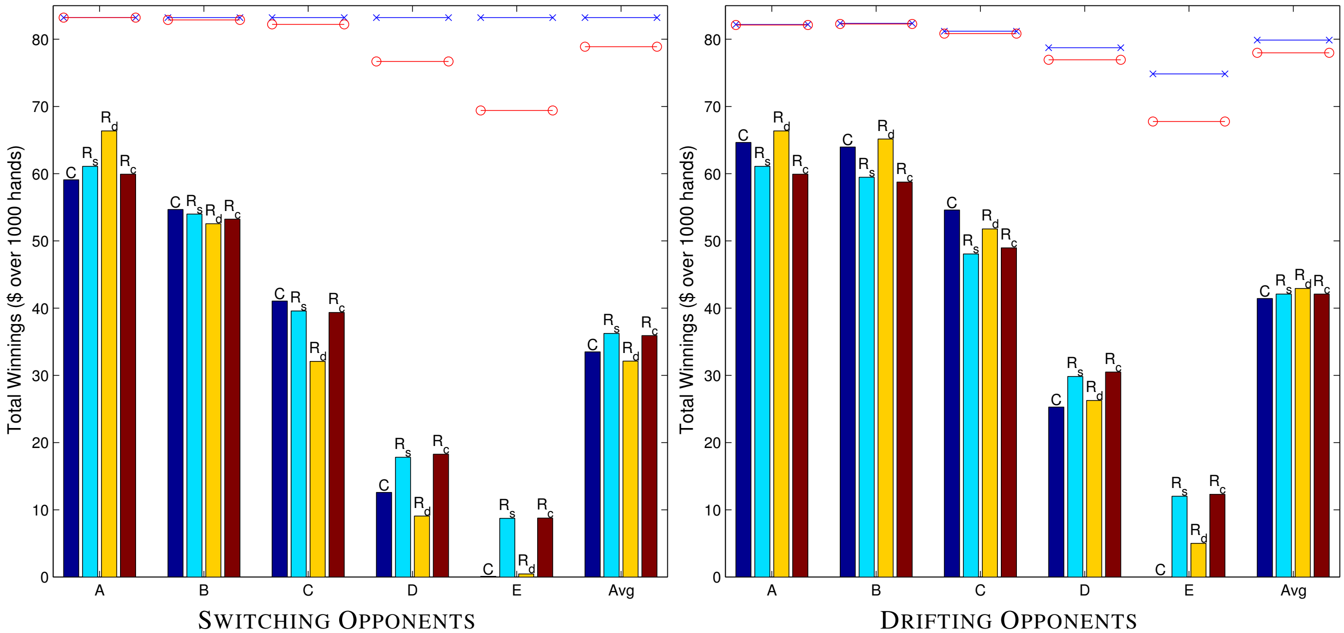

Figure 2: Total winnings of RBPF modelling against different switching opponents (left) and drifting opponents (right). The bars labelled "C" come from Figure 1.

example, when playing switching opponent D, if we incorrectly believed the opponent was stationary (model A) we would actually lose money on the match.[3]

It is interesting to note that when playing against a static opponent (opponent A) it can be beneficial to model them as if they moved with low probability or a slight drift (model B). This phenomenon is common for particle filtering and is a result of the bias that comes from using a finite number of particles (Liu & West 2000).

Unknown Motion Parameters. A more realistic setting is if the opponent's motion parameters are not known. One simple approach is to choose a fixed motion parameter that works well even when incorrect. Consulting Figure 1, model C for switching and drifting opponents has the highest average performance across the five opponents (see the bars on the far right of the graph).

Alternatively, we could use a RBPF to infer this parameter of the motion model, as described in the previous section. The results of this approach for both switching and drifting opponents is shown in Figure 2. For each opponent we show a bar for the best static model from the previous results (model C) as well as three RBPF modellers. R_s corresponds to using our prior over switching probabilities (assuming no drift), R_d corresponds to using our prior over drifting amounts (assuming no switching), and R_c corresponds to an independent prior over both forming a combined model.

Examining model C and R_s in the left-hand graph of Figure 2 we see that against most opponents R_s performs better than the static model and never performs significantly worse. Against highly switching opponent E it actually performs quite a bit better than the static model as it can infer that its opponent is switching rapidly. On average, although the improvement is not dramatic, it is still suggesting that the inference is providing some benefit. We see a similar result against drifting opponents (comparing model C and R_d).

Taking matters one step further, we are not likely to have even the form of our opponent's motion correct. We can observe the effect of this error in Figure 2. Consider model R_d in the left-hand graph and model R_s in the right-hand graph. This shows the result of using a completely different form to model our opponent's motion. Although the model performs worse than the correct model[4], the loss is never more than \$9 over 1000 hands. This is relatively small considering the incorrect beliefs. More importantly, the average performance using our dual estimation with the wrong form is very similar to model C — the best average static model with the correct form from Figure 1. Hence, dual estimation may even compensate for using the wrong form of the motion model.

Unknown Model. If even the form of the model is unknown, but a small set of possible forms can be enumerated, we can still employ dual estimation. R_c in Figure 2 shows the results of a combined prior. The performance is reasonably competitive with the correct model against all op-

[3]Although not shown, the loss is almost \$22 for 1000 hands. This is our worst case result, which is still considerably higher than the equilibrium value of the game of –\$55.

[4]Note that R_d actually does worse than R_s at modelling drift opponents with a high drift standard deviation. This is likely due to the fact that for our choice of prior, the switching model actually assigns a higher likelihood than the drift model to large strategy changes.

ponents suggesting that even this lack of knowledge can be accounted for with dual estimation.

Conclusion

In this paper we advocated the use of state estimation techniques as a promising approach for modelling agents with dynamic behavior. We described the application of particle filtering to the domain of Kuhn poker, and showed that it could effectively model and exploit both static and dynamic opponents. We also explored the use of a dual estimation technique for estimating both the opponent's strategy and dynamics model, demonstrating significant additional robustness to incorrect beliefs.

Although this work has focused on the small domain of Kuhn poker, we believe the techniques can be effectively applied to much larger domains. Particle filters can be used effectively in a large domain as long as agent behavior can be represented with a relatively small number of parameters and the motion and observation models can be computed efficiently. Although the size of the Texas Hold'em game tree is intractably large, we are investigating parameterizations of the full game that might be sufficient for describing and exploiting a substantial range of human play. In addition, it has already been shown that observation models can be computed even for the full game of Texas Hold'em in real-time (Southey *et al.* 2005).

In addition to applying these ideas to the full game, we are also planning on investigating more interesting forms of agent dynamics. Strong human players do not adapt their play in a random fashion nor are they oblivious to recent play. In the future we hope to investigate the effectiveness and robustness of such dynamics models that may more accurately capture likely human dynamics.

Acknowledgments

We would like to thank Rob Holte and Bret Hoehn for their valuable insights, along with all of the members of the University of Alberta Computer Poker Research Group. This research was supported by NSERC, iCore, and Alberta Ingenuity through the Alberta Ingenuity Centre for Machine Learning.

References

Bererton, C. 2004. State estimation for game ai using particle filters. In *AAAI workshop on challenges in game AI*.

Billings, D.; Papp, D.; Schaeffer, J.; and Szafron, D. 1998. Opponent modeling in poker. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 493–498. Madison, WI: AAAI Press.

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale po ker. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

Billings, D.; Davidson, A.; Schauenberg, T.; Burch, N.; Bowling, M.; Holte, R.; Schaeffer, J.; and Szafron, D. 2004. Game tree search with adaptation in stochastic imperfect information games. In *Computers and Games*.

Bui, H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the abstract hidden markov model. *Journal of Artificial Intelligence Research* 17:451–499.

Carmel, D., and Markovitch, S. 1996. Learning models of intelligent agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Davidson, A.; Billings, D.; Schaeffer, J.; and Szafron, D. 2000. Improved opponent modeling in poker. In *Proceedings of the 2000 International Conference on Artificial Intelligence*.

Doucet, A.; de Freitas, N.; and Gordon, N., eds. 2000. *Sequential Monte Carlo Methods in Practice. New York*. Springer-Verlag, New York.

Gilpin, A., and Sandholm, T. 2006. A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*.

Han, K., and Veloso, M. 1999. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the International Symposium of Robotics Research*, 199–204.

Hoehn, B.; Southey, F.; Holte, R. C.; and Bulitko, V. 2005. Effective short-term opponent exploitation in simplified poker. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 783–788.

Hoehn, B. 2006. *The Effectiveness of Opponent Modelling in a Small Imperfect Information Game*. Ph.D. Dissertation, University of Alberta.

Kautz, H. A. 1991. A formal theory of plan recognition and its implementation. In Allen, J. F.; Kautz, H. A.; Pelavin, R.; and Tenenberg, J., eds., *Reasoning About Plans*. Morgan Kaufmann Publishers. 69–125.

Kuhn, H. W. 1950. A simplified two-person poker. *Contributions to the Theory of Games* 1:97–103.

Liu, J., and West, M. 2000. Combined parameter and state estimation in simulation-based filtering. In Doucet, A.; de Freitas, N.; and Gordon, N., eds., *Sequential Monte Carlo Methods in Practice. New York*. Springer-Verlag, New York.

Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 663–670.

Southey, F.; Bowling, M.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 550–558.

Storvik, G. 2002. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing* 50:281–289.