# Opponent Modeling with POMDPs

Daniel Mescheder Karl Tuyls Michael Kaisers

*Maastricht University, P.O. Box 616, 6200 MD Maastricht*

**Abstract**

Reinforcement Learning techniques such as Q-learning are commonly studied in the context of two-player repeated games. However, Q-learning fails to converge to best response behavior even against simple strategies such as Tit-for-two-Tat. Opponent Modeling (OM) can be used to overcome this problem. This article shows that OM based on Partially Observable Markov Decision Processes (POMDPs) can represent a large class of opponent strategies. A variation of McCallum's *Utile Distinction Memory* algorithm is presented as a means to compute such a POMDP opponent model. This technique is based on Baum-Welch maximum likelihood estimation and uses a t-test to adjust the number of model states. Experimental results demonstrate that this algorithm can identify the structure of strategies against which pure Q-learning is insufficient. This provides a basis for best response behavior against a larger class of strategies.

## 1 Introduction

Games commonly serve as a model to study effects and behaviors that can be observed in economy, society and technology. A famous instance is the Prisoner's Dilemma (PD) which will serve as the main example throughout this paper: Two criminals are interrogated independently by a police officer. Each of them has the option to "defect", i.e. to incriminate his companion or to "collaborate". Depending on their answers, their sentences will be more or less severe (see [1] for details). The PD is widely used to explain price setting in an economic duopoly.

In the field of Multi-Agent Learning (MAL), games provide a framework for analyzing the behavior of learning techniques. In the PD well known learning algorithms as for example Q-learning have turned out to converge to the Nash equilibrium "always defect" in self-play. It will be demonstrated, in how far this is a suboptimal outcome and how Opponent Modeling (OM) can lead to better performance.

Over the years, scholars have proposed several OM algorithms. The *fictitious play* algorithm introduced by Brown in 1951 [3] and Tesauro's *Hyper-Q* [11] calculate the average of the opponent's actions as an estimator of her current mixed strategy. Carmel and Markovitch developed the *US-L** algorithm which creates an Opponent Model based on Moore Automata [4]. The predictions of fictitious play and Hyper-Q are expected to be reasonable against strategies that are stationary or vary only insignificantly. However, a strategy such as Tit-for-two-Tat cannot be modeled this way. In contrast, US-L* can model opponents playing deterministic strategies but countering a stochastic policy the identified automaton will become unfeasibly large. We will show that opponent models based on POMDPs constitute a compromise between these two extremes. To compute a POMDP Opponent Model (POMDPOM), this paper will introduce a variation of the *Utile Distinction Memory* approach by McCallum [9].

The following section provides several preliminary definitions and a notion of optimality. Section 3 will present one of the main contributions of this paper which is to make the link between OM and the study of POMDPs. In Section 4, an algorithm to compute such OMs is presented. Section 5 provides experimental results. Finally, the conclusions are drawn.

## 2 Background

This section provides fundamental definitions and preliminaries for the upcoming discussion. Firstly, the notions of games and strategies are introduced; secondly the concept of Reinforcement Learning is outlined and finally a notion of optimality in games is presented.

## 2.1 Games and Strategies

This paper will concentrate on iterated stateless two-player games at the example of the PD. The definitions used in this paper mostly follow the formalisms used by Carmel and Markovitch [4] and extend them by allowing mixed (i.e. stochastic) strategies.

**Definition 1.** *A* two-player game *is a tuple $G = (A_0, A_1, u_0, u_1)$ where $A_i = \{0, \ldots, n_{a_i}\}$ is the set of actions available to player $i$ and $u_i : A_0 \times A_1 \to \mathbb{R}$ is player $i$'s utility function [4].*

The utility describes the desirability of each pair of actions to a player. An example for such a game is the Prisoner's Dilemma described in the introduction. Its utility matrix is given by:

| | collaborate | defect |
|---|---|---|
| collaborate | 3, 3 | 0, 5 |
| defect | 5, 0 | 1, 1 |

Here the first value is the row player's rewards and the second one is the reward for the column player. A new class of games can be created by repeating a two-player game infinitely or indefinitely often. The repeated version of the PD game will be referred to as the Iterated Prisoner's Dilemma (IPD). The list of joint actions played up to stage $n$ shall be called a *history* of length $n$ and $\mathcal{H}$ will be the set of all possible histories. A strategy $S_i : \mathcal{H} \times A_i \to [0, 1]$ is a function which maps from a history of a game to a probability distribution over all pure actions, or equivalently to a *mixed action*.

**Definition 2.** *A* two-player repeated game *based on a stage game $G$ is a tuple $G^{\#} = (G, S_0, S_1, U_0, U_1)$ where $S_i$ is the set of all strategies for player $i$ and $U_i : S_0 \times S_1 \to \mathbb{R}$ is the expected long term utility of playing $S_0$ against $S_1$ for player $i$.*

For the repeated game, the utility functions should capture the idea of long term effects of an action. A popular choice is the expected sum of discounted rewards of a pair of strategies with a discount factor $0 \leq \gamma < 1$ which controls the trade-off between immediate rewards and long term gains [5] The function $U_i^{\#}$ shall denote the expected utility after $n$ steps. This relates to the utility given in Definition 2 as $U_i(S_0, S_1) = U_i^{\#}(0, S_0, S_1)$.

The main goal of the learning algorithms presented later in this paper can roughly be described as optimizing the strategy with respect to this notion of utility. Before elaborating this further, some special cases for strategies shall be presented.

**Definition 3.** *A* stationary strategy *is a strategy where the probability of choosing an action does not depend on the current history [4]. $S$ is stationary iff $\forall h_1, h_2 : S(h_1, a) = S(h_2, a)$.*

**Definition 4.** *A* Moore Machine *is a tuple $M = (Q, q_0, \Sigma, \Lambda, \delta, o)$ where $Q$ is a set of states, $q_0$ is the initial state, $\Sigma$ is the alphabet of input symbols, $\Lambda$ is the alphabet of output symbols, $\delta : Q \times \Sigma \to Q$ is the state transition function and $o : Q \to \Lambda$ is the output function that associates every state with an output symbol.*

A strategy which can be computed by a Moore Machine will be called a *deterministic finite strategy*. Tit-for-Tat is a strategy for the IPD which has especially become popular after the experiments performed by Axelrod [1]. A Tit-for-Tat playing agent will collaborate in the first move and then copy his opponent's previous action in all subsequent steps. Tit-for-n-Tat is a generalization of this idea: An agent following this strategy will keep collaborating and allow the opponent to defect $n - 1$ times. Only if the opponent has defected $n$ or more times in a row, the Tit-for-n-Tat agent will defect as well.

Every Tit-for-n-Tat strategy can be represented by a Moore Machine. Let the set of actions be $A = \Sigma = \Lambda = \{0, 1\}$ where 0 stands for "defect" and 1 stands for "collaborate". Then a machine that computes a Tit-for-n-Tat strategy is given by $M_n = (Q, q_0, \Sigma, \Lambda, \delta, o)$ with $Q = \{0, 1, \ldots, n\}$, $q_0 = 0$, and the following transition and output functions:

$$\delta(s, a) = \begin{cases} \min(s + 1, n) & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases} \qquad o(s) = \begin{cases} 0 & \text{if } s = n \\ 1 & \text{otherwise} \end{cases}$$

Throughout this paper, Tit-for-n-Tat strategies will be used as an example for deterministic finite strategies.

## 2.2 Reinforcement Learning

The problem of acting optimally in a completely observable environment exhibiting the Markov Property has been largely solved thanks to Reinforcement Learning (RL). One of the most popular RL-algorithms is Q-learning [10]. Q-learning seeks to approximate the utility of every pure action in each state.

**Definition 5.** *Let $\alpha$ denote the learning rate, $\gamma$ the discount factor and $r_\tau$ be the reward observed in the $\tau$'th step. then the following is called the* Q-learning update rule *for a state $s$ and an action $a$:*

$$Q_{\tau+1}(s,a) \leftarrow Q_\tau(s,a) + \alpha\left(r_\tau + \gamma \max_{a'} Q_\tau(s,a') - Q_\tau(s,a)\right)$$

In the case of stateless games, the state information is irrelevant. A Q-learning agent in this setting will simply update the quality estimate for the previously played action. Q-Learning in a stateless game can be considered as a function $\mathcal{L}_{\alpha,\gamma} : \mathcal{H} \to (A \to \mathbb{R})$ that associates with each history of the game a mapping from actions to quality estimates.

**Definition 6.** *An* action selection scheme *is a function $\rho : A \times (A \to \mathbb{R}) \to \mathbb{R}$ that maps from a function assigning utility estimates to actions and an action to a probability of selecting that action.*

A popular example is the Boltzmann action selection shown in Equation 1. The parameter $t$ is used to balance between the exploration of new strategies and the exploitation of previously gained knowledge [10].

$$\rho(a,Q) = \frac{e^{Q(a)\cdot t^{-1}}}{\sum_{a' \in A} e^{Q(a')t^{-1}}} \tag{1}$$

It is easy to see that a Q-learning function $\mathcal{L}$ and an action selection scheme $\rho$ together form a strategy which will be referred to as the *Q-learning strategy*.

## 2.3 Notions of Optimality

A primary concern will be to find an optimum in the space of strategies of a repeated game $G^\#$. In this section this idea will be formalized. It will be shown how stationary strategies, deterministic finite strategies and Q-learning fit into this notion and why it is sensible to consider Opponent Modeling.

In the preceding section the notion of utility in multi stage games was defined. This concept gives rise to the idea of a best-response strategy.

**Definition 7.** *A strategy $S_0$ is called a* best response strategy *against, $S_1$ with respect to a set of strategies $\mathbb{S}$, iff $\forall S' \in \mathbb{S} : U_0(S', S_1) \leq U_0(S_0, S_1)$.*

As an example, it can easily be shown that in the IPD the strategy "always defect" is a best response against any stationary strategy. The expected reward of a defecting agent playing against a stationary strategy with a chance of collaborating denoted by $y$ is given by $U_0(S_0, S_1(y)) = \sum_{i=0}^{\infty}(1-y)\gamma^i + 5y\gamma^i = \frac{1+4y}{1-\gamma}$. Furthermore, if the discounted reward sum utility measure is used with $\gamma \geq \frac{1}{2}$ then Tit-for-Tat is a best response against itself [5, p. 111] whose expected reward is $\frac{3}{1-\gamma}$. In MAL it is generally not very interesting to consider best response strategies as such, as the focus of interest is on learning policies. Such a strategy may not play optimally at first, but very well in the long run. In order to substantiate this notion of "long run optimality", the concept of a limit best response strategy will be introduced:

**Definition 8.** *A strategy $S_0$ will be called the* limit best response *against $S_1$ with respect to a set of strategies $\mathbb{S}$ iff $\forall S' \in \mathbb{S}, \epsilon > 0 \exists n \geq 0 : U_0^\#(n, S', S_1) < U_0^\#(n, S_0, S_1) + \epsilon$.*

Obviously, any strategy which is a best response against $S_1$ with respect to $\mathbb{S}$ will also be a limit best response against $S_1$ with respect to $\mathbb{S}$. As an example, consider the Q-learning strategy described in Section 2.2. Countering a stationary strategy is equivalent to acting in a Markov Decision Process with just one state. The reward of an action $a$ is a random variable only depending on $a$. Tsitsiklis proved that in such a setting Q-learning converges to the optimal value function $Q^\star$ [12]. This relies on some assumptions, most of which are trivially met in the case under consideration as no noise is involved and there is just one state. Thus, if the action selection policy $\rho$ converges to greedy behavior over time, then Q-learning is a limit best response against any stationary strategy. However, its behavior against strategies which do take into account the history of the game is expected to be less satisfactory. Consider the Tit-for-two-Tat strategy.

Carmel and Markovitch proved that for every finite deterministic strategy there exists a finite deterministic best response [4]. In the case of Tit-for-two-Tat, this best response consists in alternately defecting and collaborating. Q-learning cannot find this best response behavior as information such as "defecting once leads to collaboration; defecting twice leads to a defect" is discarded and only immediate rewards are taken into account.

Previous research furthermore investigated the behavior of Q-learning in self-play. It has been shown that the game tends to converge to stationary subgame-perfect equilibria [8]. In the IPD, it was demonstrated that Q-learning agents converge towards the strategy "always defect". From the calculation above it is known that the utility of playing "always defect", against an always defecting agent, i.e. the equilibrium that Q-learning converges to is $\frac{1}{1-\gamma}$. Thus, Tit-for-Tat expects a three times higher longterm reward. At the same time Tit-for-Tat versus Tit-for-Tat is a stable pairing of strategies, as no player can improve by changing the strategy. A good learning algorithm for MAL should find such an equilibrium in self-play.

Consider the scenario where the learning agent is given a description of her opponent. This would turn the problem at hand into a discrete optimal control problem in which the opponent is the system which is to be controlled. Scholars in control theory have discussed several classes of systems for which there exist optimal or at least approximately optimal control algorithms. As, however, such a model is generally not given in advance, it is necessary to first identify the parameters of the underlying model with the help of the opponents input/output behavior before such a technique can be used.

The following section will provide an insight which leads to a new class of opponent modeling algorithms based on POMDPs.

# 3 Link to POMDPs

In the following it will be assumed that the opponent possesses a set of internal states. Her current mixed action depends only on her current state, i.e. every internal state is associated with a probability distribution over all pure actions. Furthermore, it is assumed that the opponent will transfer to another internal state during each iteration and the probability of transferring to a certain state only depends on her current state and the action she observed. This in fact is equivalent to the Markov Property and the class of agents described in this paragraph are those strategies which can be represented by a Partially Observable Markov Decision Processes (POMDP).

**Definition 9.** *A* Partially Observable Markov Decision Process *(POMDP) is a tuple* $(\mathcal{Q}, q, \Sigma, \Lambda, \Delta, \mathcal{O})$ *where* $\mathcal{Q}$ *is a set of states,* $\Sigma$ *is a set of input symbols,* $\Lambda$ *is a set of output symbols (observations),* $\Delta : \Sigma \times \mathcal{Q} \times \mathcal{Q} \to [0,1]$ *is a transfer function where* $\Delta(x, s, s_\tau)$ *is the probability of transferring to state* $s$ *given that the current state is* $s_\tau$ *and input* $x$ *was observed.* $\mathcal{O} : \Lambda \times \mathcal{Q} \to [0,1]$ *is an observation function where* $\mathcal{O}(y, s_\tau)$ *denotes the probability of seeing output* $y$ *given that the current state is* $s_\tau$*. Finally,* $p(s)$ *is the initial probability of state* $s$*.*

There are several reasons why this limiting assumption is reasonable. Firstly, the problem Carmel and Markovitch considered [4], the identification of Moore Machines, is a special case of the identification problem of POMDPs:

**Theorem 1.** *Every Moore Machine is also a POMDP.*

*Proof.* By construction: Let $(Q, q_0, \Sigma, \Lambda, \delta, o)$ be a Moore Machine. Then an equivalent POMDP with state-space $\mathcal{Q} = Q$ is given by:

$$p_i = \begin{cases} 1 & \text{if } i = q_0 \\ 0 & \text{otherwise} \end{cases} \quad \Delta(x, i, j) = \begin{cases} 1 & \text{if } \delta(j, x) = i \\ 0 & \text{otherwise} \end{cases} \quad \mathcal{O}(i, j) = \begin{cases} 1 & \text{if } o(j, x) = i \\ 0 & \text{otherwise} \end{cases}$$

□

Thus, a learning algorithm which is able to identify POMDPs is expected to play successfully against deterministic finite strategies. It is also easy to see that every stationary strategy can be represented by a POMDP: Let $\pi(a_i) = P(A = a_i)$ be the probability of playing a pure action $a_i$. Then an equivalent POMDP with just one state can be constructed by taking $p(s) = 1$, $\Delta(x, s, s') = 1$ and $\mathcal{O}(i, j) = \pi(a_i)$.

Even though in practice it is probably not possible to identify general POMDPs with infinite state spaces, it is interesting to note that if one allows for this generalization, POMDPs encompass the set of all strategies defined in Section 2.1.

**Theorem 2.** *Every strategy $S$ as given in Definition 2 can be represented by a POMDP with an at most countably infinite number of states.*

This can be proven intuitively by letting the state space of the POMDP be $\mathcal{H}$ and the transitions reflect the action probabilities of the strategy. As the set of all histories is countably infinite, for every strategy there exists a POMDP with a countably infinite number of states. Consequently, also the Q-learning strategy can be represented by such a POMDP.

Up to this point, the opponent has been treated as an entity separate from her environment. An alternative perspective is to view the opponent as an *unobservable part of the environment* whose new state space is the Cartesian product of the environment's original state space and the agent's internal state space. The idea of embedding an opponent model into the state space resembles the I-POMDP approach presented by Gmytrasiewicz and Doshi [6]. However, in the I-POMDP framework, each state corresponds to a complete possible opponent model whilst the POMDPOM approach relies on including the opponent's internal state into the environment's state space.

From this perspective, and with the restrictions on the opponent presented above, the problem of playing optimally against an opponent becomes equivalent to the problem of acting optimally in a partially observable environment. Several techniques have been proposed for learning in partially observable environments. However, it appears that none of them guarantees an optimal result. The aim of the following section will be to approximately identify a POMDP, i.e. to compute the transition- and observation functions that best explain the observed behavior. Once such a model has been identified, the result will be not much different from a regular MDP. In fact, we can construct an MDP from the POMDPOM simply by "forgetting" about the observation function. Therefore, once such a model has been exhibited, RL techniques as Q-learning (Definition 5) constitute a means to achieve optimal control.

# 4 Computing a POMDPOM

In this section, an algorithm shall be presented that can be used for computing POMDPOMs. The goal is to identify the underlying POMDP from the observed input/output behavior. The techniques presented here are based on the work by McCallum on the topic of Reinforcement Learning with aliased states [9].

**Baum-Welch Maximum Likelihood Estimation** A POMDP under a given input sequence of length $n$ can be regarded as a probability distribution over all possible output sequences of length $n$. Consider the case where the number of states of the underlying POMDP is known or can be estimated. Then the task of deriving a set of parameters for a POMDP that can best explain the observed sequence is actually a maximum likelihood estimation.

Baum and Welch proposed a hill-climbing method for maximum likelihood estimations in Hidden Markov Models (HMM) [2]. This method can be modified so that it also works for POMDPs. The Baum-Welch procedure has been proven to converge to a local maximum. At the time of writing, there is no algorithm that guarantees convergence to a globally optimal solution.

**State Splitting** In general, the number of states of a POMDP cannot be known a priori. The function *update-pomdp* outlined in Algorithm 1 seeks to overcome this problem and to add states dynamically by statistical evidence. The function is called after a fixed number of steps with a fixed length history $h$. It consists of an outer and an inner iteration. The inner iteration uses the Baum-Welch procedure to find a maximum likelihood model given the current observations and the current number of states.

The outer loop is responsible for determining the correct number of states. It uses the current model and the input/output data to create confidence intervals over the expected output following each transition. If the confidence intervals of two transitions leading to the same state do not overlap, it can be concluded with statistical significance, that the output of that particular state depends on the history of the game. That, however, violates the Markov property. The conflict is solved by adding a new state for each cluster of overlapping confidence intervals. This last step can be shown to be an instance of the minimum clique cover problem from graph theory which is NP-complete. Yet, a number of approximation algorithms and heuristics exist [7].

McCallum used a similar approach and introduced the idea of splitting states based on the expected responses associated with state transitions [9]. However, instead of constructing confidence intervals over

**Algorithm 1** Updating the POMDP model

```
1:  function UPDATE-POMDP(xs, ys, m)        ▷ xs: inputs, ys: observations, m: current model
2:      repeat                               ▷ While new states are being added
3:          l ← LOG-LIKELIHOOD(xs, ys, m)
4:          repeat                           ▷ Until the likelihood converges
5:              l_old ← l
6:              m ← BAUM-WELCH(xs, ys, m)
7:              l ← LOG-LIKELIHOOD(xs, ys, m)
8:          until |l − l_old| < ε
9:          m_old ← m
10:         c ← CONFIDENCE-CLUSTER(xs, ys, m)   ▷ Find overlapping confidence intervals
11:         m ← SPLIT-STATES(c, m)              ▷ New model according to cluster
12:     until |m| = |m_old|
13:     return m
```

the opponent's outputs, he tested for statistically significant differences between the expected discounted future rewards of each pair of transitions.

The presented procedure circumvents the main limitation of pure Baum-Welch updates: The number of states does not have to be known in advance anymore. Nevertheless, some approximations and heuristics are required. It is thus not obvious, how well these techniques will perform in practice. The following section will present a number of quantitative experiments to assess the quality of each of the identification mechanisms presented above.

# 5 Experiments and Results

In the last section a means of constructing POMDPOMs was introduced. In this section, the performance of pure Baum-Welch maximum likelihood estimation and the state-splitting procedure will be assessed.

**Performance of the Baum-Welch Procedure** The performance of the pure Baum-Welch procedure was tested on a set of deterministic finite strategies. This is to examine how well it performs if the target machine is indeed a POMDP, but the size of the state-space is estimated incorrectly.

For the experiment a large sample of the opponent's behavior is generated, i.e. a set of input sequences together with their corresponding output sequences. Subsequently, in each learning step a Baum-Welch update is performed with the current input/output sequence. The *agreement* between the original strategy and the model is the measured as the ratio of the samples which the model can predict correctly.

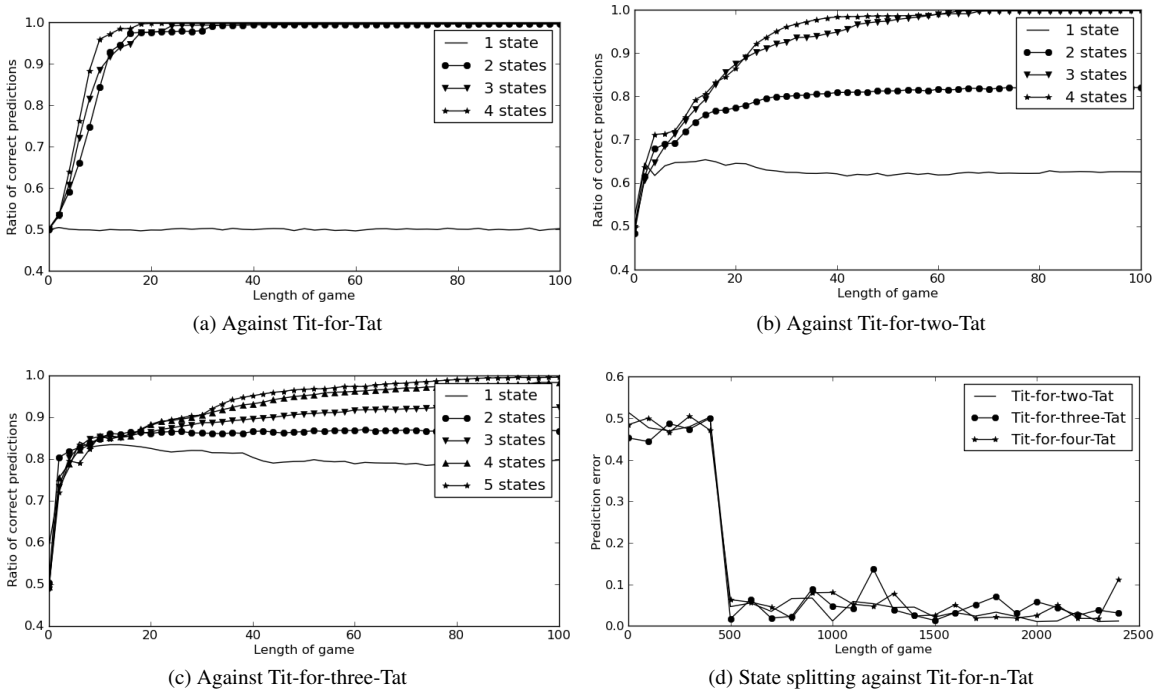
During the experiment, the learning agent plays a stationary strategy with 50% probability for each action. The first test is performed with Tit-for-Tat. The graph in Figure 1a shows that for all models with at least two states, the Baum-Welch procedure converges to the correct set of parameters.

Similar results are found for the identification of a Tit-for-two-Tat player (Figure 1b). Against a Tit-for-three-Tat strategy (Figure 1c) a much slower convergence can be noted. Nevertheless the correct model is learned eventually if the number of states is chosen adequately. Yet, even models with three states yield good results: A three-state model predicted nearly 90% of the sample correctly. This indicates, that for some applications modeling with a lower than optimal state-space might still perform satisfactorily.

**Performance of the State Splitting Procedure** The state splitting identification procedure is tested by looking at the prediction error of the model: In each step, the modelling agent is asked to make a prediction about the current mixed strategy of her opponent. The absolute difference between the real mixed strategy and the prediction is then plotted as the current discrepancy between the model and the real strategy.

In Figure 1d the state splitting procedure with 500 sample steps and a significance $\sigma = 0.1$ is tested against Tit-for-n-Tat strategies. The results show that for none of the opponent strategies, optimal convergence of the model can be guaranteed. However, already after the first learning period, the average error decreases considerably.

Figure 2a shows the prediction error in an IPD against Q-learning agents with varying learning rates. The general prediction error is lower than the one observed in the case of Tit-for-n-Tat strategies. However, this




(a) Against Tit-for-Tat

(b) Against Tit-for-two-Tat

(c) Against Tit-for-three-Tat

(d) State splitting against Tit-for-n-Tat

Figure 1: Agreement between Tit-for-n-Tat strategies and the learned model measured by the number of correct predictions on a sample of the original agent's behavior. In Figure 1d the difference between the opponent's mixed strategy and the corresponding estimate is measured.

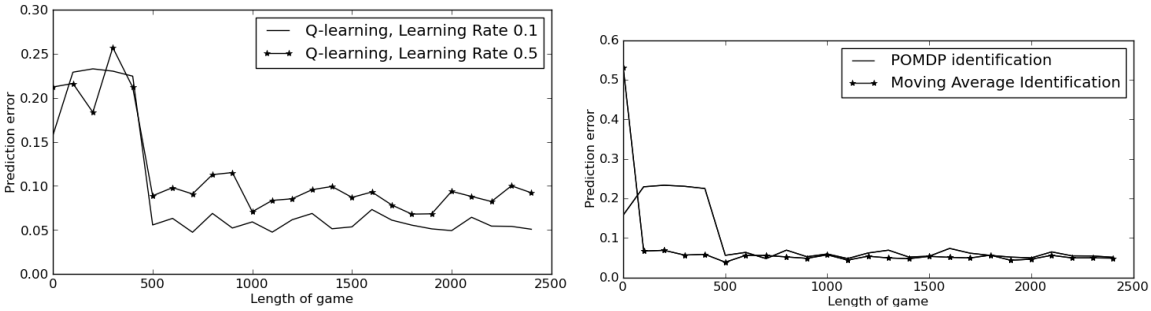
is mainly because in contrast to deterministic finite strategies, Q-learning agents will play a mixed strategy. Therefore if the model errs in its prediction, the implications will be less severe. Consequently, it is difficult to compare these results with each other. Furthermore, even though the model does predict the Q-learner's action reasonably well, it would be wrong to conclude, that a representation of Q-learning is detected in the process. In fact, during most sample runs, no state splitting takes place and the resulting model is still a simple two-state POMDPOM. Figure 2a also shows why this can be problematic: A higher learning rate (which leads to a more "jumpy" behavior of the Q-learner) makes it more difficult for the model to generate the right prediction.

In spite of the underfitting problem in the case of Q-learning the quantitative results are generally encouraging: Figure 2b shows that the POMDP-based estimate is not significantly worse than the averaging procedure that fictitious play bases on. At the same time it performs well against discrete finite strategies - a domain in which the moving average-based approach is insufficient.

# 6 Conclusion and Discussion

This paper investigated Opponent Modeling as a means to approach best response behavior. In the first part, the concept of limit best response strategies was introduced. It was shown that OM constitutes a basis for limit best response behavior. Moreover, POMDPs are a promising framework as they encompass a large class of strategies. The algorithm presented was transferred from the domain of learning with aliased states to the domain of OM. Results show that this technique can indeed approximate the internal structure of an opponent. However, it makes use of some heuristics: Firstly, the Baum-Welch procedure which does not guarantee convergence to a global optimum and secondly the method which is used for clustering in the state splitting procedure. Therefore, it is difficult to provide theoretical guarantees for this approach.

Furthermore, the Utile Distinction Memory algorithm was merely tested on its ability to predict the opponent's behavior. The original algorithm used by McCallum [9] integrates the resulting prediction with Q-learning to actually learn an optimal strategy. It will be interesting to see how such a modeling and learning agent competes against the various common strategies that were examined in this paper. Moreover, the problem of balancing exploration and exploitation becomes interesting in the light of OM. For the results




(a) Identification of Q-learning

(b) POMDP Identification vs Moving Average

Figure 2: Performance of state splitting vs Q-learning. Figure 2a shows the prediction error for varying learning rates. Figure 2b compares the POMDP identification with the prediction error of a moving average.

shown in Section 5, the modeling agent played a uniform stationary strategy. It is conceivable that better results would arise, if the agent for instance tried to explicitly collect data about one particular state of the opponent model in order to determine whether it would be reasonable to split it up.

Experimental evidences suggest that the algorithm is able to predict the behavior of Tit-for-n-Tat and to some degree the more complex Q-learning strategy. Yet, the result for the latter is still not fully satisfactory, as the model fails to capture the precise dynamics of Q-learning. Compared to other OM algorithms, the POMDP learner seems to be a viable compromise. It performs reasonably well in the domain of deterministic finite strategies but also approximates the behavior of stochastic players.

# References


[1] R. Axelrod. *The evolution of strategies in the iterated prisoners dilemma*, pages 32–41. 1987.

[2] LE Baum, T Petrie, and G Soules. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 1970.

[3] G. W. Brown. *Iterative solution of games by fictitious play*, pages 374–376. Wiley, 1951.

[4] David Carmel and Shaul Markovitch. Model-based learning of interaction strategies in multi-agent systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 10(3):309–332, July 1998.

[5] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[6] Piotr J. Gmytrasiewicz and Prashant Doshi. A Framework for Sequential Planning in Multi-Agent Settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[7] Jens Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction, exact, and heuristic algorithms for clique cover. In *Proceedings of the eighth Workshop on Algorithm Engineering and Experiments and the third Workshop on Analytic Algorithmics and Combinatorics*, volume 123, page 86. Society for Industrial Mathematics, 2006.

[8] Michael Kaisers and Karl Tuyls. Frequency adjusted multi-agent Q-learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 309–316. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[9] Andrew McCallum. *Reinforcement learning with selective perception and hidden state*. Phd thesis, University of Rochester, 1996.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning : An Introduction*, volume 3. MIT Press, September 1998.

[11] Gerald Tesauro. Extending Q-learning to general adaptive multi-agent systems. *Advances in neural information processing systems*, 16, 2004.

[12] John N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3):185–202, September 1994.