POMDP Opponent Models for Best Response Behavior

Daniel Mescheder

June 24, 2011

Abstract

This work focuses on the integration of Opponent Modeling (OM) in two player repeated games. For this purpose, a theoretical analysis of strategies and best response strategies is given. This analysis explains why OM can be a key to long-term best response behavior. It is shown, that Opponent Models based on Partially Observable Markov Decision Processes (POMDPs) generalize a broad range of existing OM techniques. As an instance of a POMDP based learning algorithm, a variation of McCallum's *Utile Distinction Memory* algorithm is presented. This technique is based on Baum-Welch maximum likelihood estimation and uses a t-test to adjust the number of model states. Experimental results demonstrates that this algorithm can identify the structure of some popular simple strategies as for example Tit-for-Tat. It is also able to approximate the behavior of more complex examples like Q-learning.

1 Introduction

Games frequently 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 which will serve as one of the main examples 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 name Iterated Prisoner's Dilemma (IPD) will be used to refer to a variant of this game where the same scenario is repeated an indefinite number of times. Felegyhazi et al. described one particular real world application of games like the IPD [12]: In ad-hoc networks, each node can decide to either route packages or to block them, preferring his own traffic. The rewards associated with this scenario resemble the ones for the IPD. Furthermore, the Prisoner's Dilemma is widely used to explain price setting in an economic duopoly [22].

In the field of Multi-Agent Learning (MAL), games frequently serve as a framework for analyzing the behavior of learning techniques in multi-agent environments. Shoham et al. pointed out that there are several motivations for studying the behavior of agents in a MAL setting [23]. One of them is the *computational perspective*, the interest to computationally find characteristics of a game at hand, as for example equilibria. The *descriptive perspective* refers to the analysis of characteristics of learning algorithms in a MAL setting. Finally, there is the *prescriptive perspective* which aims at finding well performing strategies. This paper will mostly focus on the prescriptive approach.

Well known learning algorithms as for example Q-learning have turned out to converge to the Nash equilibrium "always defect" in the IPD (see Section 3). It will be demonstrated, in how far this is a suboptimal outcome and how Opponent Modeling (OM) can lead to better performance. In particular if it is assumed that the opponent's internal dynamics base on a Partially Observable Markov Decision Process (POMDP). A lot of literature treats the problem of optimal learning in partially observable environments. This paper will highlight a variant of the *Utile Distinction Memory* approach by McCallum [20, 21] as a particular example. It is shown experimentally, that this technique indeed can be used as an OM algorithm.

For this purpose, the following section will provide several preliminary definitions. Subsequently, a theoretical analysis of the concept of optimality in games will be presented which also motivates the use of OM. Section 4 introduces and analyzes previous results on this problem and Section 5 builds a link between OM and the study of POMDPs. In the subsequent section, the algorithms which will be investigated in this paper are presented and Section 7 provides experimental results. Finally, a conclusion will be drawn and the findings will be discussed.

2 Preliminaries

Before turning to the analysis of optimality in games, 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.

2.1 Games and Strategies

This paper will concentrate on iterated stateless two-player games, mostly at the example of the IPD, even though some of the finding can probably be generalized to broader frameworks as for example Markov Games [19]. This section seeks to formalize the notion of iterated two-player games. The definitions used in this paper mostly follow the formalisms used by Carmel and Markovitch [8] 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* $A_i = \{0, \dots, n_{a_i}\}$ *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 [8].*

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

	collaborate	defect
collaborate	3, 3	0, 5
defect	5, 0	1, 1

Another popular example for a two player game is *Battle of the Sexes*. In this game, a couple wants to spend time together. This can happen either at the opera or at a football match. If they fail to meet, neither player receives a reward. If, however, both decide for the football match, the man will get a slightly higher payoff than the woman, who in turn will be better off, if the date takes place at the opera. The payoff matrix for this game looks as follows:

	opera	football
opera	3, 2	0, 0
football	0, 0	2, 3

A new class of games can be created by repeating a two-player game infinitely or indefinitely often. $h = \left((a_0^{(0)}, a_1^{(0)}), (a_0^{(1)}, a_1^{(1)}), \dots, (a_0^{(n-1)}, a_1^{(n-1)})\right)$ shall be called a *history* of a game of length n and $\mathcal{H}$ will be the set of all histories. The empty history shall be denoted by $\emptyset$. Moreover, the notation $h : (a_0, a_1)$ shall denote the concatenation operation that yields the history h followed by player 0 playing a_0 and player 1 playing a_1.

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 from a history 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 for U_i is the expected sum of discounted rewards of a pair of strategies:

$$U_i(S_0, S_1) = E\left[\sum_{j=0}^{\infty} \gamma^j r_i^{(j)}\right]$$

where $r_i^{(j)}$ is the reward player i receives j steps into the future and $0 \le \gamma < 1$ is a discount factor which controls the trade-off between immediate rewards and long term gains [19].

Using the notion of discounted reward sums, a recursive scheme resembling the Bellman equation can be derived. Let $U_i^{\mathcal{H}}$ denote the expected reward sum for player i, given a previous history h. This relates to the utility described in definition 2 by

$$U_i(S_0, S_1) = U_i^{\mathcal{H}}(\emptyset, S_0, S_1)$$

There exists a recursive definition of $U_i^{\mathcal{H}}$:

$$U_i^{\mathcal{H}}(h, S_0, S_1) = \sum_{\substack{(a_0,a_1)\in \\ A_0\times A_1}} S_0(h, a_0) S_1(h, a_1) u_i(a_0, a_1) + \gamma U_i\left(h : (a_0, a_1), S_0, S_1\right)$$

Furthermore the function $U_i^\#$ shall denote the expected utility of a given stage n.

$$U_i^\#(n, S_0, S_1) = \sum_{|h|=n} U_i^{\mathcal{H}}(h, S_0, S_1) \cdot p(h, S_0, S_1)$$

where $p(h, S_0, S_1)$ is the probability of a history h under the two strategies S_0 and S_1. p can be computed recursively:

$$p\left(h_s : (a_0, a_1), S_0, S_1\right) = S_0(h_s, a_0) S_1(h_s, a_1) p(h_s, S_0, S_1)$$

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 is independent from the current history [7].* 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,* Σ *is the alphabet of input symbols,* Λ *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:

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

$$o(s) = \begin{cases} 0 & \text{if } s = n \\ 1 & \text{otherwise} \end{cases}$$

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 [25, 28]. Q-learning seeks to approximate the utility of every pure action in every state.

Definition 5. *Let α denote the learning rate and γ the discount factor. Let furthermore r_τ be the reward observed in the τ'th step. then the following is called the* Q-learning update rule *for a state s and an action a:*

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

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 Softmax or Boltzmann action selection [25, 16]:

$$\rho(a,Q) = \frac{e^{Q(a)\cdot\tau^{-1}}}{\sum_{a' \in A} e^{Q(a')\tau^{-1}}}$$

Another technique commonly used is the ϵ-greedy approach [25]:

$$\rho(a,Q) = \begin{cases} 1-\epsilon & \text{if } a = \text{argmax}_{a'} Q(a') \\ \epsilon & \text{otherwise} \end{cases}$$

The constants ϵ and τ are parameters which can be used to balance between the exploration of new strategies and the exploitation of previously gained knowledge.

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

$$S_q(h,a) = \rho(a, \mathcal{L}_{\alpha,\gamma}(h))$$

Making use of the above concepts, the next section will present theoretical results about the question, how optimal play can be achieved in two-player repeated games.

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 the notion of optimality shall be formalized. Subsequently it will be shown how stationary strategies, deterministic finite strategies and Q-learning fit into this notion and finally it will be argued why it is sensible to consider Opponent Modeling as a means to reach or at least to get close to optimality.

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 [18].

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 consider the set of all stationary strategies in the IPD:

Theorem 1. *In the IPD, the strategy "always defect" is a best response against any stationary strategy with respect to the set of stationary strategies.*

Proof. Let S_1 be a stationary strategy. Let y denote the probability of playing "collaborate" under S_1. Let furthermore S_0 be the stationary strategy which is sought to be optimal against S_1 and let x denote the probability of playing "collaborate" under S_0. It will be shown that $x = 0$ maximizes the discounted reward sum U_0. Let the expected future reward of playing S_0 with x be denoted as $r(x)$. Then by definition of the discounted reward

$$\begin{aligned} r(x) &= \sum_{i=0}^{\infty} \gamma^i \left(3xy + 5(1-x)y + 1(1-x)(1-y)\right) \\ &= \sum_{i=0}^{\infty} \gamma^i \left(-3xy + 4y + x + 1\right) \end{aligned}$$

The derivative of this function with respect to x is thus

$$\frac{dr}{dx} = \sum_{i=0}^{\infty} \gamma^i \left(-3y + 1\right)$$

Which means that r is strictly monotonic decreasing iff $y > -\frac{1}{3}$, which is always the case as $y \in [0, 1]$. Thus a minimal value of x maximizes r. Knowing that $x \in [0, 1]$, the best response to any stationary strategy is to play "collaborate" with a probability of zero, i.e. to always defect. □

The expected reward of a defecting agent playing against a stationary strategy with a chance of collaborating denoted by y is given by

$$U_0(S_0, S_1(y)) = \sum_{i=0}^{\infty} (1-y)\gamma^i + 5y\gamma^i = \frac{1+4y}{1-\gamma} \quad (1)$$

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 as 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 $Q^\star$ [27]. 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 these assumptions are met and if the action selection policy ρ converges to greedy behavior over time, then Q-learning is a limit best response against any stationary strategy with respect to the set of all stationary strategies.

Limit best response behavior is a relaxation of the concept of a best response strategy. One should, however, keep in mind that this notion in its most general form does not lead to a practicable framework for real world algorithms, as the following theorem shows:

Theorem 2. *No strategy which is computable by a finite algorithm can be a limit best response in every game against every strategy and with respect to the set of all strategies.*

Proof. Consider the Battle of the Sexes game described above. Let S_0 be any strategy which is computable by a finite algorithm. For any finite algorithm there exists a Turing Machine computing the same function. Let M_0 be the Turing Machine computing S_0. For every Turing Machine M it is possible to construct an second Turing Machine T which simulates M and performs further calculations with the result [24].

This can be used to construct a new Turing Machine $M^\star$ computing the strategy function $S^\star$ shown in Algorithm 1. Obviously, for any value of n, $U_0^{\#}(n, S_0, S^\star) = 0$. The way $M^\star$ is constructed guarantees that the couple always fails to meet.

Algorithm 1 The new Turing Machine $M^\star$

```
1: On input $h$
2: Simulate $M_0$ on $h$
3: if Result of $M_0$ is "opera" then
4:     return "football"
5: if Result of $M_0$ is "football" then
6:     return "opera"
```

At the same time, $U_0^{\#}(n, S^\star, S^\star) \geq 2$ for every n and every discount factor γ, as both players will always meet and the immediate reward will always be at least the reward of the least favored option.

However, this means that for every S_0 there exists a value ϵ, a strategy S' and an opponent strategy $S_2 \in \mathbb{S}$, where $\epsilon = 1$, $S' = S^\star$ and $S_2 = S^\star$, such that for all n,

$$U_0^{\#}(n, S', S_1) > U_0^{\#}(n, S_0, S_1) + \epsilon$$

Thus, no computable strategy can satisfy the general limit best response criterion. □

This suggests that in order to get any guarantees, it is reasonable to restrict the class of opponents in question and/or to limit the set of strategies an algorithm is compared to.

The results above indicate that the Q-learning strategy performs well compared to stationary strategies. 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[8]. 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 on the behavior of Q-learning in self-play. It has been shown that the

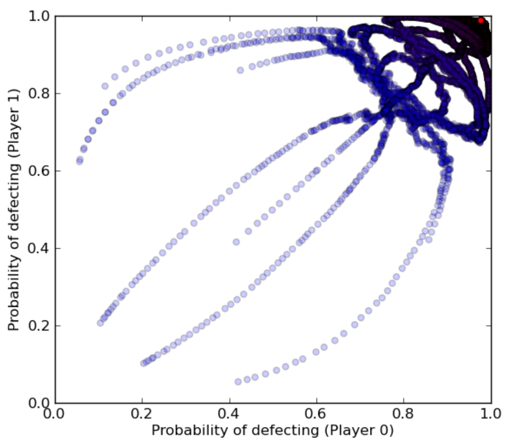

Figure 1: Traces of two Q-learning agents with Boltzmann action selection in the IPD, $\tau = \frac{1}{100}$, $\alpha = \frac{1}{100}$, $\gamma = 0$. The strategies converge towards the Nash equilibrium "always defect".

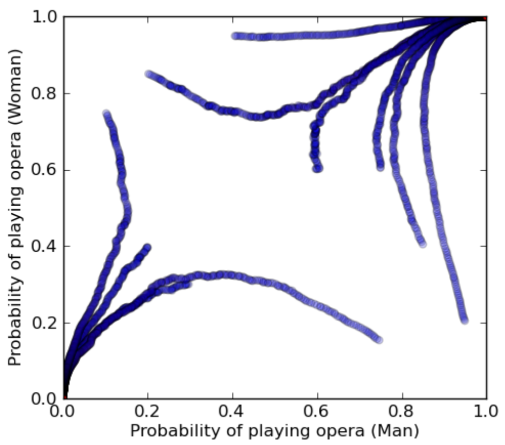

Figure 2: Traces of two Q-learning agents with Boltzmann action selection in the Battle of the Sexes game, $\tau = \frac{5}{100}$, $\alpha = \frac{1}{1000}$, $\gamma = 0$. Depending on the starting mixed strategies, the game converges either to the equilibrium "always play football" or to "always play opera".

game tends to converge to stationary subgame-perfect equilibria [16, 17]. Figure 1 shows the traces of two Q-learning agents in the IPD. The traces of Q-learning in the Battle of the Sexes game (Figure 2) furthermore indicate that in this game Q-learning is not a limit best response to itself: It seems that a better strategy than the behavior shown would be to insist on ones preferred choice (opera or football) and to let the other player adapt.

Also the learning result in the IPD is not satisfactory, as there exists a pair of strategies which form a Nash equilibrium (i.e. they are mutual best responses) whose reward is higher than the reward obtained with Q-learning, as the following, well known result shows:

Theorem 3. *If $\gamma \geq \frac{1}{2}$, then Tit-for-Tat is a best response strategy to Tit-for-Tat in the IPD*[1] *(see [13, p. 111]).*

For this result to hold it is important that the stage game is repeated infinitely or indefinitely often. The expected reward of a Tit-for-Tat agent playing against Tit-for-Tat is

$$U_0(S_0, S_1) = \sum_{i=0}^{\infty} 3 \cdot \gamma^i = \frac{3}{1-\gamma}$$

From equation 1 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 $\frac{1}{1-\gamma}$. In other words, Tit-for-Tat expects a three times better longterm reward. At the same time this pairing of strategies is stable, as no player can improve by changing the strategy. A good learning algorithm for MAL should find such an equilibrium in self-play.

It is reasonable to assume that for every strategy that the learning agent will ever encounter, there exists a finite description. Note that the proof of Theorem 2 still covers this case, i.e. it is still not possible to achieve general limit best response behavior with a finite algorithm. 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 and optimal control is achieved if the opponent produces actions that are desirable from the learners perspective. 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 next section describes some of the approaches that have been discussed in literature in the context of Opponent Modeling.

4 Related Work

Over the years scholars have proposed several OM techniques. The *fictitious play* algorithm introduced by Brown in 1951 [4, 3] can already be seen as an OM technique. A player following this strategy creates a model of her opponent by averaging all the past outputs she observed. The result is an estimate of the opponent's current mixed strategy. The player will then use her knowledge by playing the best response pure action against this mixed strategy. Tesauro proposed a similar but more sophisticated algorithm named *Hyper-Q* which uses a weighted average of the opponent's actions as an estimator for her current mixed strategy [26]. It is assumed that the opponent's internal state coincides with

[1] In fact it can be shown that this pair of strategies is even subgame perfect, i.e. no player can gain by deviating from the strategy in any subgame.

this very strategy but in contrast to Brown's fictitious play, it is not assumed that the "greedy", immediate best response leads to the highest future payoff. Instead the mixed strategy is fed as the current state into Q-learning which then serves as a quality estimator.

The predictions from fictitious play and Hyper-Q are expected to be reasonable against stationary players and against strategies that vary only insignificantly during each learning step as for example Q-learning with a low learning rate. However, a player based on a Moore Machine such as Tit-for-two-Tat cannot be modeled using this approach.

Carmel and Markovitch developed an Opponent Model based on Moore Automata [6, 7, 8]. The idea is to generate an automaton which explains and generalizes the past observations. Finding a minimal automaton for a sequence of observations has been shown to be NP-complete and is thus not feasible in practice [14]. Carmel and Markovitch propose a heuristic called *US-L** which bounds the size of the model by

$$|M_{k+1}| = |M_k| + |h| + 1$$

where $|M_{k+1}|$ is the size of the new model, $|M_k|$ is the size of the old one and $|h|$ is the length of the current game history, i.e. the number of outputs that have been observed.

In this article, only the case will be considered in which just one sequential experiment can be performed, i.e. that the opponent can not be "reset" to an initial state such that the game starts over from the beginning. Under this restriction, US-L* turns out not to be practicable.

Algorithm 2 Identification algorithm for Moore Machines. M is the current model, i is the last input and r is the response that followed the input i.

```
1: function update-model($M, i, r$)
2:     $(Q, q_0, \Sigma, \Lambda, \delta, o) \leftarrow M$
3:     $Q' \leftarrow Q \cup \{\max(Q) + 1\}$
4:     $\delta'(s,a) = \begin{cases} \max(Q)+1 & \text{if } s = \max(Q) \wedge a = i \\ \max(Q)+1 & \text{if } s = \max(Q)+1 \\ \delta(s,a) & \text{otherwise} \end{cases}$
5:     $o'(s) = \begin{cases} r & \text{if } s = \max(Q)+1 \\ o(s) & \text{otherwise} \end{cases}$
6:     return $(Q', q_0, \Sigma, \Lambda, \delta', o')$
```

Theorem 4. *If the identification is based on just one sequential experiment, then there exists an algorithm which bounds the opponent model by*

$$|M_{k+1}| = k + 2 = |h|$$

Proof. Consider the update-mechanism shown in algorithm 2. Moreover, let the initial machine be given by $Q = \{0\}$, $q_0 = 0, \delta(s,a) = 0$ and $o(s) = r_0$ where r_0 is the opponent's response to the empty input. After every update, max Q is the model state which is reached by following the current history. In every step exactly one state is added which corresponds to the newly observed output. The bound clearly holds for the initial automaton where $|M_0| = |h_0| = 1$. Let $|M_k| = k + 1$, exactly one state is added during the model update, thus $|M_{k+1}| = k + 2$. □

Theorem 7 in appendix A demonstrates, that US-L* does perform worse than Algorithm 2 for certain examples. Thus, for the case of a sequential experiment, a simple algorithm as given in Theorem 4 guarantees stricter bounds than US-L*.

In general, a technique based on the identification of Moore Machines is expected to work well against opponents playing deterministic finite strategies. Countering a stochastic policy as for example Q-learning, however, the state-space of the identified automaton will become unfeasibly large.

The following section will provide an insight which leads to a new class of opponent modeling algorithms based on POMDPs which seek to combine the strengths of the methods presented in this section.

5 Link to POMDPs

It has been shown in Theorem 2, that it is not possible to derive a learning algorithm which is optimal against every type of player. It is therefore necessary to introduce a set of limiting assumptions. 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 the possible 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* $(\mathbb{A}, C, n_a, n_o, p)$ *where* $\mathbb{A} = (A^{(0)}, A^{(1)}, \ldots, A^{(n_a-1)})$*. Here* n_a *is the number of input actions which, for simplicity, are supposed to be integers between 0 and* $n_a - 1$*. Let* n_s *be the size of the state space and* n_o *be the number of pure outputs, then for all* x *and all* j*,*

$$\sum_{i=1}^{n_s} A^{(x)}_{(i,j)} = \sum_{i=1}^{n_o} C_{(i,j)} = \sum_{i=1}^{n_s} p_i = 1 \qquad (2)$$

Let $P(X(\tau) = x)$ *denote the probability of observing input action* x *in step* τ*. Let furthermore* $P(S(\tau) = s)$ *be the probability that the POMDP is in state* s *at time* τ *and* $P(Y(\tau) = y)$ *the probability that the POMDP produces the output* y *at time* τ*. Then the* *interpretation of A, C and p is that*

$$\begin{aligned}
A^{(x)}_{(s,s_\tau)} &= P\left(S(\tau+1) = s \mid X(\tau) = x, S(\tau) = s_\tau\right) \\
C_{(y,s_\tau)} &= P\left(Y(\tau) = y \mid S(\tau) = s_\tau\right) \\
p_s &= P\left(S(\tau) = s\right)
\end{aligned}$$

There are several reasons why this limiting assumption is reasonable. Firstly, the problem Carmel and Markovitch considered [8], the optimal control of Moore Machines, is a special case of the optimal control problem of POMDPs:

Theorem 5. *Every Moore Machine is also a POMDP.*

Proof. By construction: Let $(Q, q_0, n_a, n_o, \delta, o)$ be a Moore Machine. Then an equivalent POMDP with state-space $Q' = Q$ is given by:

$$p_i = \begin{cases} 1 & \text{if } i = q_0 \\ 0 & \text{otherwise} \end{cases}$$

$$A^{(x)}_{(i,j)} = \begin{cases} 1 & \text{if } \delta(j,x) = i \\ 0 & \text{otherwise} \end{cases}$$

$$C_{(i,j)} = \begin{cases} 1 & \text{if } o(j,x) = i \\ 0 & \text{otherwise} \end{cases}$$

$\square$

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 $p = (1)$, $A^{(x)} = [1]$ and $C_{(i,j)} = \pi(a_i)$.

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.

Theorem 6. *Every strategy S as given in Definition 2 can be represented by a POMDP with an at most countably infinite number of states.*

Proof. By construction. Let the state space of the POMDP be $\mathcal{H}$. Let

$$p_i = \begin{cases} 1 & \text{if } i = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$A^{(x)}_{(h_i:(a_i,a_j)),h_j)} = \begin{cases} S(h_j, a_i) & \text{if } h_i = h_j \wedge a_j = x \\ 0 & \text{otherwise} \end{cases}$$

$$C_{(a,h)} = \begin{cases} S(a, \emptyset) & \text{if } h = \emptyset \\ 1 & \text{if } h = hs : (a_i, a_j) \wedge a_i = a \\ 0 & \text{otherwise} \end{cases}$$

As the set of all histories is countably infinite, for every strategy there exists a POMDP with a countably infinite number of states. $\square$

Corollary 1. *The Q-learning strategy can be represented as a POMDP with a countably infinite number of states.*

Up to this point, the opponent has always 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. 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. In the next section, two approximation techniques to identify POMDP based opponents will be presented.

6 Learning algorithms based on POMDPs

In this section, two learning algorithms based on POMDPs shall be presented. The goal will be to explicitly identify the underlying POMDP from a sequence of observed input/output behavior. The techniques presented here are based on the work by McCallum on the topic of Reinforcement Learning with aliased states [21, 20]. Consider a strategy which is indeed based on a POMDP and whose model can be identified. In that case RL techniques as Q-learning (Definition 5) constitute a means to achieve optimal control. Therefore, an adequate POMDP based identification technique is expected to eventually lead to limit best response behavior with respect to the class of POMDP-based strategies.

The first technique presented below introduces the limitation that the number of internal states of the opponent is known or can be estimated. If this is the case, a Baum-Welch based maximum likelihood estimation can be performed to improve the parameters of the POMDP such that the model explains the data observed as accurately as possible.

The second technique builds on the first algorithm and introduces an outer loop that performs statistical tests to determine whether or not it is reasonable to increase the number of states of the model. This reduces the limitation of the first algorithm at the price of a higher computational complexity.

6.1 Continuous 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). This method can be modified so that it also works for POMDPs. A detailed description is given in appendix B. 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.

6.2 Maximum Likelihood and State Splitting

In general, the number of states of a POMDP cannot be known a priori. The function *update-pomdp* outlined in Algorithm 3 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. While the outer loop is responsible for determining the correct number of states, the inner iteration seeks to find a maximum likelihood model given the current observations and the current number of states.

This inner loop makes use of the Baum-Welch method used in Section 6.1 and described in Appendix B. The Baum-Welch update is applied repeatedly until the log-likelihood improvement falls below a threshold ϵ. At this point the model is assumed to have converged.

Algorithm 3 Updating the POMDP model

```
1:  function update-pomdp($xs, ys, m$)
2:      repeat                          ▷ While new states are being added
3:          $l \leftarrow$ log-likelihood($xs, ys, m$)
4:          repeat                      ▷ Until the likelihood converges
5:              $l_{old} \leftarrow l$
6:              $m \leftarrow$ baum-welch($xs, ys, m$)
7:              $l \leftarrow$ log-likelihood($xs, ys, m$)
8:          until $|l - l_{old}| < \epsilon$
9:          $m_{old} \leftarrow m$
10:         $c \leftarrow$ confidence-cluster($xs, ys, m$)
11:         $m \leftarrow$ split-states($c, m$)
12:     until $|m| = |m_{old}|$
13:     return $m$
```

The outer loop 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. A detailed description of the state-splitting algorithm is given in appendix C.

This procedure bases loosely on the perceptual distinctions approach by Chrisman [10]. McCallum used a similar approach and introduced the idea of splitting states based on the expected responses associated with state transitions [20, 21]. However, instead of constructing confidence intervals over the opponent's outputs, he tested for statistically significant differences between the expected discounted future rewards of each pair of transitions.

With the state splitting procedure described above, a technique was introduced which 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 had to be used. It is thus not obvious, how well these techniques will perform in practice. The following section will present a number of quantitative experiments to asses the quality of each of the identification mechanisms presented above.

7 Experiments and Results

In the last section two algorithms were introduced as a means of identifying opponent strategies. In this section these two techniques will be tested on their predictive performance.

7.1 Performance of the Baum-Welch Procedure

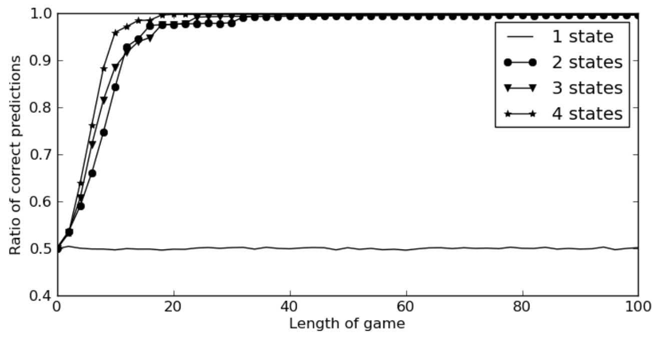

Figure 3: Agreement between Tit-for-Tat strategy and the learned model measured by the number of correct predictions on a sample of the original agent's behavior.

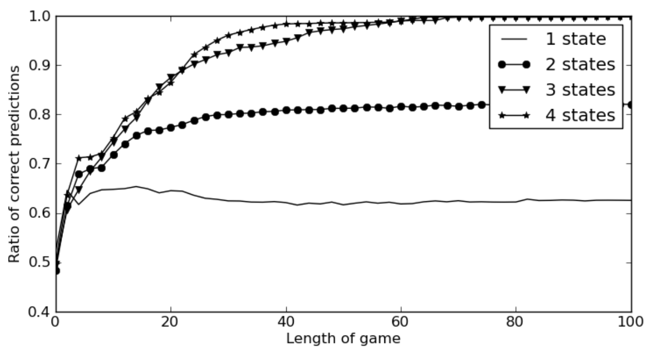

Figure 4: Agreement between Tit-for-two-Tat strategy and the learned model

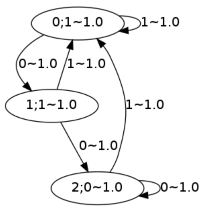

Figure 5: Model of the Tit-for-two-Tat strategy learned by the Baum-Welch procedure. The states are labeled with $i : o \sim p$ where i is the name of the state, o is an output and p is the probability of observing o in state i. The transitions are labeled as $a \sim p$ where a is the action which triggers the transition at hand and p is the transition probability given that a was played.

The performance of the pure Baum-Welch procedure was tested on a set of deterministic finite strategies. This is to examine how well the procedure presented in appendix B 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 every 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 3 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 4). An example of a resulting model with three states is shown in Figure 5. The test with the Tit-for-three-Tat strategy (Figures 6 and 7) reveal a much slower convergence. Nevertheless the correct model was learned eventually if the number of states was chosen

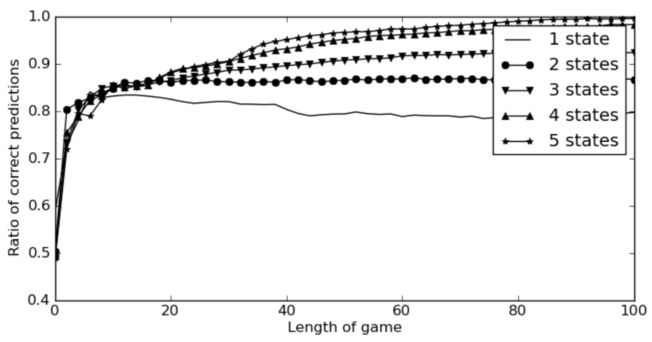

Figure 6: Agreement between Tit-for-three-Tat strategy and the learned model.

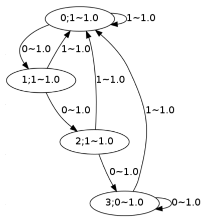

Figure 7: Model of the Tit-for-three-Tat strategy learned by the Baum-Welch procedure

adequately. Yet, even models with three states yielded 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.

7.2 Performance of the State Splitting Procedure

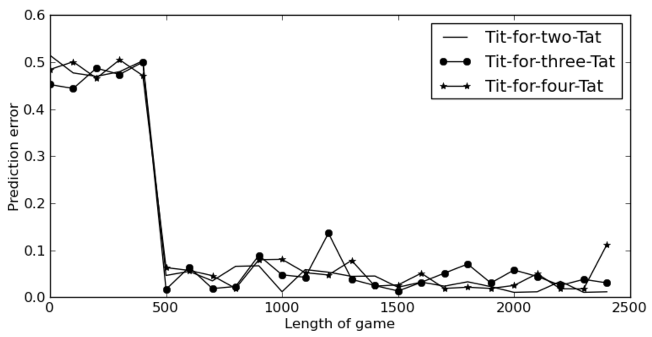

Figure 8: Error of the model prediction against Tit-for-n-Tat strategies in the IPD.

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

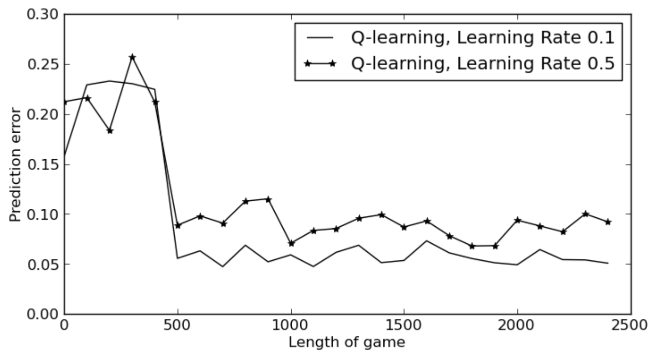

Figure 9: Prediction error of the model of two Q-learning agents in the IPD with different learning rates

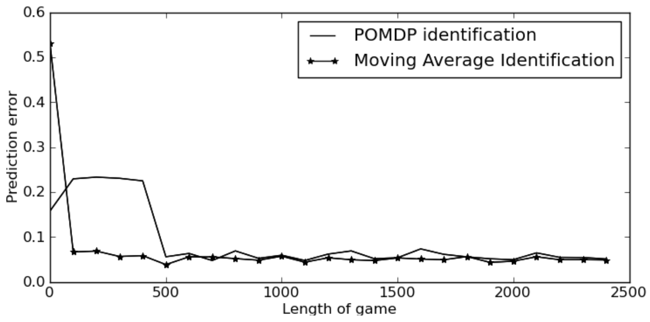

Figure 10: Prediction error of the POMDP identification algorithm compared to the error of a prediction based on the moving average of past observations in a IPD game against Q-learning with learning rate 0.1.

model and the real strategy.

In Figure 8 the state splitting procedure with 500 sample steps and a significance $\sigma = 0.1$ was 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 9 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 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 was detected in the process. In fact, during most sample runs, no state splitting took place and the resulting model was still simply based on a maximum likelihood estimation using a two-state POMDP. Figure 9 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 10 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.

8 Conclusion and Discussion

This paper investigated on Opponent Modeling as a means to learn best response behavior. In the first part the concept of limit best response strategies was introduced. It was shown that OM constitutes a technique which might eventually lead to limit best response behavior against a large class of opponent strategies. 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. It turned out 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.

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 the algorithms presented in section 4, 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 strategies.

In this paper, the Utile Distinction Memory algorithm was merely tested on its ability to predict the opponent's behavior. The original algorithm used by McCallum [20] 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 shown in section 7, 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 model in order to determine whether it would be reasonable to split it up.

References

[1] R. Axelrod. *The evolution of strategies in the iterated prisoner's 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] U Berger. Brown's original fictitious play. *Journal of Economic Theory*, 135(1):572–578, July 2007.

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

[5] D. Carmel and S. Markovitch. Learning models of opponent's strategy in game playing, 1993.

[6] D. Carmel and Shaul Markovitch. Learning models of intelligent agents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 62–67, 1996.

[7] David Carmel and Shaul Markovitch. Opponent modeling in multi-agent systems. In *Adaption And Learning In Multi-Agent Systems*, pages 40–52. Springer-Verlag, 1995.

[8] 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.

[9] M Chen, J Li, W Li, and L Wang. Some approximation algorithms for the clique partition problem in weighted interval graphs. *Theoretical Computer Science*, 381(1-3):124–133, August 2007.

[10] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the National Conference on Artificial Intelligence*, pages 183–183. Citeseer, 1992.

[11] Pierre a Devijver. Baum's forward-backward algorithm revisited. *Pattern Recognition Letters*, 3(6):369–373, December 1985.

[12] M. Felegyhazi, J.-P. Hubaux, and L. Buttyan. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(5):463–476, May 2006.

[13] Drew Fudenberg and Jean Tirole. Game Theory, 1991.

[14] E Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, June 1978.

[15] 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.

[16] Michael Kaisers. *Reinforcement Learning in Multi-agent Games: A value iteration perspective*. Bachelor thesis, Maastricht University, 2008.

[17] 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.

[18] Kevin Leyton-brown and Yoav Shoham. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. 2010.

[19] M.L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, page 163. Citeseer, 1994.

[20] Andrew McCallum. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 190–196. Citeseer, 1993.

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

[22] Andrew Schotter. *Microeconomics: A Modern Approach*. Addison Wesley, 3rd edition, 2000.

[23] Y Shoham, R Powers, and T Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, May 2007.

[24] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, second edition, 1996.

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

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

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

[28] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992.

[29] Lloyd R Welch. Baum-Welch Algorithm. *IEEE Information Theory Society Newsletter*, 53(4), 2003.

A Shortcomings of US-L*

In Section 4 the US-L* algorithm proposed by Carmel and Markovitch [8, 5, 6] was introduced. Furthermore, an algorithm was presented which is able to identify an automaton M from a history h such that in any step k the size of the model is bounded by $|M_{k+1}| = |h|$. This section will present a proof that US-L* performs worse than that.

Theorem 7. *When US-L* is applied in the special case of a sequential experiment, then there exist examples for which it yields a model with $|M_{k+1}| > |h|$.*

Proof by counterexample. Consider the automaton with the following transition table:

	1	2	output
0 (initial)	0	1	1
1	1	0	2

The algorithm previously identified the automaton M_k with the following transition table:

	2	1	output
(1, 2)	(2, 1)	(2, 2)	2
(2, 1)	(1, 2)	(2, 1)	1
(2, 2) (initial)	(1, 2)	(2, 1)	2

This automaton supports the sequence of past observations ($\lambda \rightarrow 1$), ($2 \rightarrow 2$), ($2, 1 \rightarrow 2$) Consider the new observation ($2, 1, 2 \rightarrow 1$). The resulting tableau will look as follows:

	λ
λ	1
2	2
2, 1	2
2, 1, 2	1
1	1
2, 2	1
2, 1, 1	1
2, 1, 2, 2	1
2, 1, 2, 1	2

The upper part entries stem from the actual observations. The lower part entries have been filled up using the current automaton model. The algorithm will continue with the consistency loop during which λ is found to conflict with $(2, 1, 2)$ because

$$T((2), \lambda) = 2 \neq T((2, 1, 2, 2), \lambda) = 1$$

however, $T(\lambda, \lambda) = T((2, 1, 2), \lambda)$. According to the algorithm, this inconsistency is solved by adding a new column to the tableau and filling it with observations, ties or the current predictions of our model (precedence in that order):

	2	λ
λ	2	1
2	1	2
2, 1	1	2
2, 1, 2	1	1
1	2	1
2, 2	2	1
2, 1, 1	2	1
2, 1, 2, 2	2	1
2, 1, 2, 1	2	2

Still the table is not consistent as the entry 2 conflicts with $(2, 1)$ because $T((2, 2), 2) = 2 \neq T((2, 1, 2), 2) = 1$, but $T((2), 2) = T((2, 1), 2) = 1$. The new tableau after the update looks as follows:

	2	λ	2, 2
λ	2	1	1
2	1	2	2
2, 1	1	2	1
2, 1, 2	1	1	2
1	2	1	1
2, 2	2	1	1
2, 1, 1	2	1	1
2, 1, 2, 2	2	1	1
2, 1, 2, 1	2	2	1

This table has become consistent, but it is not yet closed as there exists no $s \in S$ such that $\text{row}((2, 1, 2, 1)) = \text{row}(s)$. The algorithm closes the tableau by moving the entry $(2, 1, 2, 1)$ to S:

	2	λ	2, 2
λ	2	1	1
2	1	2	2
2, 1	1	2	1
2, 1, 2	1	1	2
2, 1, 2, 1	2	2	1
1	2	1	1
2, 2	2	1	1
2, 1, 1	2	1	1
2, 1, 2, 2	2	1	1
2, 1, 2, 1, 2	1	2	2
2, 1, 2, 1, 1	2	1	1

This new tableau is consistent and closed, meaning it allows the construction of a new automaton [8]. The resulting machine has five states while the input consists of merely four observations. The transition table is given below:

	2	1	output
(2, 2, 1)	(1, 2, 2)	(2, 1, 1)	2
(1, 2, 2)	(2, 1, 1)	(1, 2, 1)	2
(2, 1, 1) (initial)	(1, 2, 2)	(2, 1, 1)	1
(1, 2, 1)	(1, 1, 2)	(2, 1, 1)	2
(1, 1, 2)	(2, 1, 1)	(2, 2, 1)	1

B Numerically Stable Baum-Welch updates in POMDPs

The Baum-Welch procedure, in its original formulation, is an algorithm for optimizing the parameters of a Hidden Markov Model (HMM) in order to improve the likelihood of a sequence of observed outputs [2, 29]. The procedure is based on Baum's forward-backward algorithm for HMM parameter estimation.

To be able to apply this technique in the context of OM, two modifications are necessary. Firstly, the original forward-backward algorithm suffers from numerical instabilities. Devijver proposed a equivalent reformulation of this procedure which works around these instabilities [11]. Secondly, both the Baum-Welch procedure and Devijver's version of the forward-backward algorithm are designed for HMMs, not for POMDPs.

This section will demonstrate, how Devijver's forward-backward algorithm can be restated for the case of POMDPs and how it then can be integrated in a adjusted version of the Baum-Welch update.

Let the POMDP be defined according to Definition 9 by $(\mathbb{A}, C, n_a, n_o, p)$. The aim of the POMDP forward-backward procedure is to find an estimator for $P\left(S(\tau) = s \mid [Y]_1^T = [y]_1^T, [X]_1^T = [x]_1^T\right)$. Using Bayes' Rule

$$\begin{aligned}
&P\left(S(\tau) = s \mid [Y]_1^T = [y]_1^T, [X]_1^T = [x]_1^T\right)\\
=&\frac{P\left(S(\tau) = s, [Y]_1^T = [y]_1^T \mid [X]_1^T = [x]_1^T\right)}{P\left([Y]_1^T = [y]_1^T \mid [X]_1^T = [x]_1^T\right)}\\
=&\frac{P\left(S(\tau) = s, [Y]_1^\tau = [y]_1^\tau, [Y]_{\tau+1}^T = [y]_{\tau+1}^T \mid [X]_1^T = [x]_1^T\right)}{P\left([Y]_1^\tau = [y]_1^\tau, [Y]_{\tau+1}^T = [y]_{\tau+1}^T \mid [X]_1^T = [x]_1^T\right)}\\
=&\frac{P\left(S(\tau) = s, [Y]_1^\tau = [y]_1^\tau \mid [X]_1^T = [x]_1^T\right)}{P\left([Y]_1^\tau = [y]_1^\tau \mid [X]_1^T = [x]_1^T\right)}\\
&\cdot\frac{P\left([Y]_{\tau+1}^T = [y]_{\tau+1}^T \mid S(\tau) = s, [X]_1^T = [x]_1^T\right)}{P\left([Y]_{\tau+1}^T = [y]_{\tau+1}^T \mid [Y]_1^\tau = [y]_1^\tau, [X]_1^T = [x]_1^T\right)}\\
=&\alpha(\tau, s) \cdot \beta(\tau, s)
\end{aligned}$$

The problem reduces thus to finding α and β which shall be called the forward estimate and the backward estimate respectively.

$$\mathcal{N}(\tau) = \frac{1}{P\left(Y(\tau) = y_\tau \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)}$$

Repeated application of Bayes' Rule and the definition of the POMDP, leads to the following recursive formulation of α:

$$\begin{aligned}
\alpha(\tau, s) =& \frac{P\left(S(\tau) = s, [Y]_1^\tau = [y]_1^\tau \mid [X]_1^T = [x]_1^T\right)}{P\left([Y]_1^\tau = [y]_1^\tau \mid [X]_1^T = [x]_1^T\right)}\\
=&P\left(S(\tau) = s \mid Y(\tau) = y_\tau, [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)\\
=&\frac{P\left(S(\tau) = s, Y(\tau) = y_\tau \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)}{P\left(Y(\tau) = y_\tau \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)}\\
=&\mathcal{N}(\tau) P\left(S(\tau) = s, Y(\tau) = y_\tau \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)\\
=&\mathcal{N}(\tau) \sum_\sigma P\left(S(\tau) = s \mid S(\tau-1) = \sigma, [X]_1^T = [x]_1^T\right)\\
&\cdot Y(\tau) = y_\tau S(\tau) = s, [X]_1^T = [x]_1^T\\
&\cdot S(\tau-1) = \sigma [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\\
=&\mathcal{N}(\tau) \sum_\sigma P\left(S(\tau) = s \mid S(\tau-1) = \sigma, X(\tau) = x_\tau\right)\\
&\cdot P\left(Y(\tau) = y_\tau \mid S(\tau) = s\right)\\
&\cdot P\left(S(\tau-1) = \sigma \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)\\
=&\mathcal{N}(\tau) \sum_\sigma \alpha(\tau-1, \sigma) A_{(s,\sigma)}^{(x_{\tau-1})} C_{(y_\tau, s)} \qquad (3)
\end{aligned}$$

This yields a recursive policy for calculating α. The missing piece, $\mathcal{N}(\tau)$, can be calculated using the preceding recursion step for α:

$$\begin{aligned}
\frac{1}{\mathcal{N}(\tau)} =& P\left(Y(\tau) = y_\tau \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)\\
=&\sum_s P\left(S(\tau) = s, Y(\tau) = y_\tau \mid [Y]_1^{\tau-1} = [y]_1^{\tau-1}, [X]_1^T = [x]_1^T\right)\\
=&\sum_s \sum_\sigma \alpha(\tau-1, \sigma) A_{(s,\sigma)}^{(x_{\tau-1})} C_{(y_\tau, s)} \qquad (4)
\end{aligned}$$

The common term of Equations 3 and 4 can be extracted to make the process computationally more efficient:

$$\begin{aligned}
\alpha(\tau, s) &= \mathcal{N}(\tau)\gamma(\tau, s)\\
\mathcal{N}(\tau) &= \frac{1}{\sum_\sigma \gamma(\tau, \sigma)}\\
\gamma(\tau, s) &= C_{(y_\tau, s)} \sum_\sigma \alpha(\tau-1, \sigma) A_{(x,\sigma)}^{(x_{\tau-1})}
\end{aligned}$$

The result differs from the original formulation [11] merely by the fact that the appropriate transition matrix is chosen in every recursion step. It is still necessary to calculate β, which can be reduced to a similar recursion:

$$\beta(\tau,s) = \frac{P\left([Y]_{\tau+1}^{T} = [y]_{\tau+1}^{T} \mid S(\tau)=s, [X]_1^T = [x]_1^T\right)}{P\left([Y]_{\tau+1}^{T} = [y]_{\tau+1}^{T} \mid [Y]_1^\tau = [y]_1^\tau, [X]_1^T = [x]_1^T\right)}$$
$$= \mathcal{N}(\tau+1) \sum_\sigma A_{(\sigma,s)}^{(x_\tau)} C_{(y_{\tau+1},\sigma)} \beta(\tau+1,\sigma) \tag{5}$$

The base cases of the recursion given in Equations 6 and 7 follow directly from their probabilistic interpretation:

$$\gamma(1,s) = p_s \cdot C_{(y_1,s)} \tag{6}$$
$$\beta(T,s) = 1 \tag{7}$$

Using the definitions of α and β it is now possible to derive an unbiased estimator for $P\left(S(\tau)=s \mid [Y]_1^T = [y]_1^T, [X]_1^T = [x]_1^T\right)$. It's definition bears a striking resemblance to the estimator derived by Devijver [11]. Analogue to the steps Baum and Welch took to derive their update rule, the need for two more probabilities arises: $P\left(S(\tau)=s, S(\tau+1)=\sigma \mid [Y]_1^T = [y]_1^T, [X]_1^T = [x]_1^T\right)$ and $P\left(S(\tau)=s, Y(\tau)=y \mid [Y]_1^T = [y]_1^T, [X]_1^T = [x]_1^T\right)$. Luckily, α, β and $\mathcal{N}$ again can be used to compute these probabilities.

$$P\left(S(\tau)=s, S(\tau+1)=\sigma \mid [Y]_1^T = [y]_1^T, [X]_1^T = [x]_1^T\right)$$
$$= \frac{P\left(S(\tau)=s, [Y]_1^\tau = [y]_1^\tau \mid [X]_1^T = [x]_1^T\right)}{P\left([Y]_1^\tau = [y]_1^\tau \mid [X]_1^T = [x]_1^T\right)}$$
$$\cdot \frac{P\left(S(\tau+1)=\sigma \mid S(\tau)=s, X(\tau)=x_\tau\right)}{P\left(Y(\tau+1)=y_{\tau+1} \mid [Y]_1^\tau = [y]_1^\tau, [X]_1^T = [x]_1^T\right)}$$
$$\cdot P\left(Y(\tau+1)=y_{\tau+1} \mid S(\tau+1)=\sigma\right)$$
$$\cdot \frac{P\left([Y]_{\tau+2}^T = [y]_{\tau+2}^T \mid S(\tau+1)=\sigma, [X]_1^T = [x]_1^T\right)}{P\left([Y]_{\tau+2}^T = [y]_{\tau+2}^T \mid [Y]_1^{\tau+1} = [y]_1^{\tau+1}, [X]_1^T = [x]_1^T\right)}$$
$$= \alpha(\tau,s)\beta(\tau+1,\sigma) A_{(\sigma,s)}^{(x_\tau)} C_{(y_{\tau+1},\sigma)} \mathcal{N}(\tau+1) \tag{8}$$

It turns out, that the values for $\mathcal{N}$ calculated in Equation 4 can also be used to calculate the likelihood of the observed data under the current model parameters.

$$P\left([Y]_1^T = [y]_1^T \mid [X]_1^T = [x]_1^T\right) = \prod_{\tau=1}^{T} \frac{1}{\mathcal{N}(\tau)}$$

In practice the log-likelihood will be of more interest, as it's calculation is more efficient and numerically more stable:

$$\log\left(\prod_{\tau=1}^{T} \frac{1}{\mathcal{N}(\tau)}\right) = -\sum_{\tau=1}^{T} \log(\mathcal{N}(\tau))$$

The methods derived up to this point are useful to calculate state probabilities, transition probabilities and output probabilities given a model and a sequence of inputs and outputs. As Baum and Welch did in the case of HMMs, these very probabilities will now be used to derive unbiased estimators for the model's parameters. Let $\mathbb{T}_x = \{\tau \mid x_\tau = x\}$. Then

$$\sum_{\tau=\mathbb{T}_x} \frac{P\left(S(\tau)=s \mid [X]_1^T = [x]_1^T, [Y]_1^T = [y]_1^T\right)}{|\mathbb{T}_x|}$$
$$\approx P\left(S(\tau)=s \mid X(\tau)=x\right) \tag{9}$$

A similar scheme can be used to derive an estimator for the transition probabilities:

$$\sum_{\tau\in\mathbb{T}_x} \frac{P\left(S(\tau)=s, S(\tau+1)=\sigma \mid [X]_1^T = [x]_1^T, [Y]_1^T = [y]_1^T\right)}{|\mathbb{T}_x|}$$
$$\approx P\left(S(\tau)=s, S(\tau+1)=\sigma \mid X(\tau)=x\right) \tag{10}$$
$$= P\left(S(\tau+1)=\sigma \mid S(\tau)=s, X(\tau)=x\right)$$
$$\cdot P\left(S(\tau)=s \mid X(\tau)=x\right) \tag{11}$$

where $P\left(S(\tau)=s \mid X(\tau)=x\right)$ can be approximated using the estimator from Equation 9. The result is an unbiased estimator for $P\left(S(\tau+1)=\sigma \mid S(\tau)=s, X(\tau)=x\right)$. An estimator for the output probabilities can be derived accordingly, making use of the Markov property:

$$\sum_{\tau\in\mathbb{T}_x} \frac{P\left(S(\tau)=s, Y(\tau)=y \mid [X]_1^T = [x]_1^T, [Y]_1^T = [y]_1^T\right)}{|\mathbb{T}_x|}$$
$$\approx P\left(S(\tau)=s, Y(\tau)=y \mid X(\tau)=x\right)$$
$$= P\left(Y(\tau)=y \mid S(\tau)=s, X(\tau)=x\right)$$
$$\cdot P\left(S(\tau)=s \mid X(\tau)=x\right)$$
$$= P\left(Y(\tau)=y \mid S(\tau)=s\right) \cdot P\left(S(\tau)=s \mid X(\tau)=x\right)$$

Thus,

$$P\left(Y(\tau)=y \mid S(\tau)=s\right)$$
$$\approx \sum_{\tau\in\mathbb{T}_x} \frac{P\left(S(\tau)=s, Y(\tau)=y \mid [X]_1^T = [x]_1^T, [Y]_1^T = [y]_1^T\right)}{|\mathbb{T}_x|} \tag{12}$$

Equation 12 holds for every value of x. Therefore a better estimator can be derived by averaging the values over all inputs:

$$P\left(Y(\tau)=y \mid S(\tau)=s\right)$$
$$\approx \sum_{\tau=1}^{T} \frac{P\left(S(\tau)=s, Y(\tau)=y \mid [X]_1^T = [x]_1^T, [Y]_1^T = [y]_1^T\right)}{|\mathbb{T}_{x_\tau}| \cdot n_a \cdot P\left(S(\tau)=s \mid X(\tau)=x_\tau\right)} \tag{13}$$

C Clustering of Transitions According to Confidence Intervals

McCallum proposed a technique to determine whether a state should be split into two seperate states using a statistical test. The version described in McCallum's paper constructs confidence intervals over the discounted reward sum [20]. As this paper focuses on actually identifying the opponent's strategy, this section will merely consider the actual state outputs. It remains interesting for future research to investigate the impact of considering the expected rewards instead.

The first step is to identify confidence intervals over the output following each state transition. Algorithm 4 shows how this can be achieved. First, for every state transition the number of estimated visits, the estimated sum of observed outputs after this transition and the estimated sum of squared outputs after this transition are computed. This information is then used to calculate the mean expected output for each transition. Furthermore, Student's t-distribution is used to construct a confidence interval at a σ level of significance. Here, $t^n_{\sigma/2}$ denotes the t-value of a distribution with n degrees of freedom at $\sigma/2$ error probability to each side.

Algorithm 4 Generate confidence intervals over outputs after a state transition. The results L and U are matrices of lower and upper bounds respectively [20, 21].

```
1:  function transition-intervals($xs, ys, m$)
2:      $T \leftarrow$ transition-estimates($xs, ys, m$)
3:      for $\tau \in [0, \dots, |ys| - 1]$ do
4:          $y \leftarrow ys_\tau$
5:          for $(i, j, p) \in T(\tau)$ do
6:              $p$: pprobability of going from $i$ to $j$ in step $\tau$
7:              $\text{count}_{ij} \leftarrow \text{count}_{ij} + p$
8:              $\text{sum}_{ij} \leftarrow \text{sum}_{ij} + p \cdot y$
9:              $\text{sumsquares}_{ij} \leftarrow \text{sumsquares}_{ij} + p \cdot y^2$
10:     $\forall i, j : \mu_{ij} \leftarrow \frac{\text{sum}_{ij}}{\text{count}_{ij}}$
11:     $\forall i, j : s_{ij} \leftarrow \sqrt{\frac{\text{count}_{ij} \cdot \text{sumsquares}_{ij} - \text{sum}^2_{ij}}{\text{count}_{ij} \cdot (\text{count}_{ij} - 1)}}$
12:     $\forall i, j : L_{ij} \leftarrow \mu_{ij} - t^{\lfloor \text{count}_{ij} \rfloor - 1}_{\sigma/2} \cdot \frac{s_{ij}}{\sqrt{\text{count}_{ij}}}$
13:     $\forall i, j : U_{ij} \leftarrow \mu_{ij} + t^{\lfloor \text{count}_{ij} \rfloor - 1}_{\sigma/2} \cdot \frac{s_{ij}}{\sqrt{\text{count}_{ij}}}$
14:     return $L, U$
```

The result is a tuple of matrices (L, U) where L contains the lower bounds of the significance intervals for each transition and U contains the respective upper bounds.

In the next step, for every state the set of incoming transitions is examined. Each such state corresponds to a row of L and U. The goal will be to form a set of sets (clusters) of transitions leading to one particular state. In each of these clusters, the significance intervals of the respective transitions are pairwise overlapping and the union of all clusters equals the full set of incoming transitions. Furthermore the aim is to find a clustering which is minimal, i.e. which satisfies the above constraint using the least possible number of subsets.

This problem is equivalent to the so called *minimum clique cover* problem from graph theory. Algorithm 5 shows, how an adjacency matrix of a graph can be constructed in $O(n^2)$ so that the nodes are the confidence intervals and two nodes are connected by an edge if and only if their respective intervals overlap.

Algorithm 5 Take a set of intervals, given as a vector of lower bounds and a vector of upper bounds and return the adjacency matrix of a graph whose nodes are intervals and where two nodes are connected if and only if their respective intervals overlap.

```
1:  function intervals-to-graph($l, u$)
2:      $\forall i, j : T_{ij} \leftarrow (l_i \leq u_j)$
3:      $\forall i, j : A_{ij} \leftarrow (T_{ij} \wedge T^T_{ij})$
4:      return $A$
```

The minimum clique problem can be stated as follows: Given a graph G, find the minimal number of fully connected subgraphs which cover every edge of G. Unfortunately, this problem has been shown to be NP-complete. Yet, there exists a broad range of heuristics and approximation algorithms [9, 15]. In experiments it was found that for the problem at hand the relatively simple CC-Heuristic presented by Gramm et al [15] performed sufficiently well.