Bachelor's thesis at the Department of Mathematics and Computer Science, Freie Universität Berlin,

Berlin Machine Learning & Robotics Lab

Inference-based Decision Making in Games

Tim Rakowski
Matriculation number: 4161946
rakowski@mi.fu-berlin.de

Supervisor: Marc Toussaint

June 7, 2011 Berlin

Abstract

Background: Reinforcement learning in complex games has traditionally been the domain of value- or policy iteration algorithms, resulting from their effectiveness in planning in Markov decision processes, before algorithms based on regret minimization guarantees such as upper confidence bounds applied to trees (UCT) and counterfactual regret minimization were developed and proved to be very successful, too. Meanwhile remarkably simple algorithms based on likelihood maximization where found for planning in Markov decision processes, which opened up room for new research. Applying these new methods to extensive games is the focus of this thesis.

Results: We describe a generic schema for transforming an extensive game into a multi-agent partially observable Markov decision process (POMDP), derive a strategy update based on the EM algorithm and give an implementation using the hidden Markov model. Tests on a number of minimalistic games suggest that for the two-player case equilibrium strategies are found if the game has pure Nash equilibria but otherwise only the average payoffs of the two players converge to their respective values of a mixed Nash equilibrium, i.e. no equilibrium strategies are found. Further investigation showed that the algorithmic framework is general enough to facilitate the replacement of the M-step by other update procedures such as the polynomial weights algorithm (resulting in external regret minimization) or the counterfactual regret minimization method. Using the latter update, the strategies do converge.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, 7. Juni 2011

Tim Rakowski

Contents

Chapter 1

Introduction

Traditionally the problem of solving complex games has been tackled by selectively searching the game's state space: [Zer12] provided a first, albeit flawed, description of the Minimax algorithm and [SD69] described the Alpha-beta search. Methods to improve the results mainly focused on handcrafting heuristics for specific games, primarily board games such as chess and checkers. Arthur Samuel has been the first to apply reinforcement learning methods to a complex game [Sam59, Sam67] but not until 1992 did new results in this field appear [Tes92, Tes95]. Unfortunately performance guarantees were not given and seemingly this area of research did not advance over the next twelve years [LPK, Gho04], so reinforcement learning basically delivered no explanation as to why, when or how it worked in games. Only recently have ideas from regret theory carried over into the area of game solving (e.g. [WG07, Joh07]), which come with game theoretic convergence guarantees.
Meanwhile a new method for finding optimal strategies in Markov decision processes was developed, that is based purely on probabilistic inference [Att03]. A major advantage of this approach is that the learning process can be treated the same as any other statistical learning problem, which e.g. might make it easy to model games with continuous state or action variables, introduce clustering and regression to realize generalization, benefit from efficient exact and approximate inference methods, and reason about it using the tools provided by probability theory!
Interestingly, even after Tesauro successfully demonstrated the applicability of methods for planning in Markov decision process to games, the fields remained largely separate. Neither did game solvers seize the opportunity of applying the new method of [Att03] to games, nor do reinforcement learning researchers generally analyse their algorithms in the contexts of regret minimization, coordination games or correlated equilibria. For example they might be interested in applying game theoretic approaches to multi-agent games with correlated reward functions. Also, regret is a "more appropriate measure" [LPK] than optimality or speed of convergence for reinforcement learning algorithms, but guarantees are hard to come by.
Therefore we hope to be able to elucidate some of the similarities between the seemingly so separate fields.

Chapter 2

Fundamentals

In this section we will review background knowledge required to be able to follow the following sections: The first part concerns Bayesian inference in Markov Decision Processes as it provides the mathematical foundation on which our approach is based, whereas the latter part deals with basic definitions and theorems of Game Theory laying out the field of application.
We assume the reader to be somewhat familiar with probability theory, linear algebra and analysis.

2.1 Probabilistic Decision Models

Bayesian inference is a probabilistic approach to reasoning under uncertainty utilizing a graphical representation of the joint distribution of the random variables describing the domain.

2.1.1 Bayesian Networks

A Bayesian network [Pea85] is a directed acyclic graph (X, E) where $X = X_0, \ldots, X_n$ is a set of random variables and $(X, Y) \in E$ is a directed edge from X to Y implying that random variable Y depends conditionally on random variable X and X is called a parent variable of Y.
Each Node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ which, in the case of a discrete variable, may be given as a table of the conditional probabilities.
Given a Bayesian network we specify the complete joint distribution of the domain as

$$P(X_0, \ldots, X_n) = \prod_{i=0}^{n} P(X_i|Parents(X_i)).$$

According to the chain rule the joint distribution would generally be given as

$$P(X_0, \ldots, X_n) = \prod_{i=0}^{n} P(X_i|X_{i-1}, \ldots, X_0).$$

Provided that the nodes are labelled according to a topological ordering the network therefore may be interpreted as a collection of assurances of conditional independence: A random variable X_i is conditionally independent of it's non-descendants given $Parents(X_i)$.

Exact Inference in Bayesian Networks

The primary objective of specifying a Bayesian network is to have a means to calculate a probability distribution over the values of a query variable X_i given an event e - an assignment of values to some observable variables E:

$$P(X_i|e) = \alpha P(X_i, e) = \alpha \sum_{z \in Z} P(X_i, e, z)$$

Here $Z = X \setminus (\{X_i\} \cup E)$ are neither query variables nor evidence variables.

2.1.2 Markov Processes

Markov processes [Mar13] describe domains consisting of a possibly infinite sequences of variables $X_{0:t} = \langle X_i \rangle_{i=0}^{t}$ and $E_{1:t} = \langle E_i \rangle_{i=1}^{t}$ where for any time t X_t is the set of hidden variables and E_t the set of evidence variables. Assuming that the process is stationary - it does not change over time - and meets the Markov property - given the current state the future states are independent of the past states - we are given an initial unconditional probability distribution $P(X_0)$, a transition model $P(X_t|X_{t-1})$ and a sensor model $P(E_t|X_t)$.
These three distributions specify the complete joint probability distribution over all variables up to time t:

$$P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1}^{t} P(X_i|X_{i-1}) P(E_i|X_i)$$

Dynamic Bayesian Network

The dependencies we just described give rise to the notion of dynamic Bayesian networks (DBNs) [DK89]: For every time slice t we have an instance of the Bayesian net consisting of the variables X_t and E_t. The variables in X_t (for $t > 0$) depend conditionally on X_{t-1} (sparsely if possible: Not all variables in X_t need to depend on all variables in X_{t-1}).

Bayes filter

Given a Markov process we are interested in the following distributions:

$P(X_t|e_{1:t})$**:** The distribution over the states at time t given all evidences up to and including that time. The problem of calculating this is called filtering. Usually we are estimating the current state given all evidences up until then.

$P(X_{t+k}|e_{1:t}), k > 0$**:** The distribution over the states at time $t + k$ given all evidences up to time t. This is called predicting as we are calculating a distribution over future states.

$P(X_k|e_{1:t}), 0 \leq k < t$**:** The problem of calculating the distribution over the states X_k given evidences up to a time $t > k$ is called smoothing. Generally we are improving our filtering result from a past time k by considering the evidences we have gained since then.

Applying various laws of probability we find that all of them have simple recursive definitions [BP66]:

$$P(X_t|e_{1:t}) = \alpha P(e_t|X_t)\left(\sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}|e_{1:t-1})\right)$$
$$P(X_{t+k}|e_{1:t}) = \alpha\left(\sum_{x_{t+k-1}} P(X_{t+k}|x_{t+k-1})P(x_{t+k-1}|e_{1:t})\right)$$
$$P(X_k|e_{1:t}) = \alpha P(X_k|e_{1:k})P(e_{k+1:t}|X_k)$$
$$P(e_{k+1:t}|X_k) = \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})P(x_{k+1}|X_k)$$

Hidden Markov Model

One particularly simple type of Markov process is the hidden Markov model (HMM) [BP66]: The set of state variables X contains exactly one variable S and the set of evidence variables exactly one variable O. Every Markov process with hidden variables X and evidence variables E may be converted into a hidden Markov model by having S be the Cartesian product of the variables in X and O be the Cartesian Product of the evidence variables E both distributed according to the probability distributions induced by the Bayesian network.
The HMM gives rise to a particularly simple format for the inference process on the basis of linear algebra: The transition model becomes an $S \times S$ matrix T, where $T_{ij} = P(X_t = j|X_{t-1} = i)$. The observation model becomes an $S \times S$ diagonal matrix O, where $O_{ii} = P(e_t|X_t = i)$. The so called *forward messages* $P(X_t|e_{1:t})$ are stored in vectors $f_{1:t}$, while the *backward messages* $P(e_{k+1:t}|X_k)$ are stored in $b_{k:t}$. Thereby we can express the filtering equations as follows:

$$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$
$$b_{k:t} = T O_t b_{k+1:t}$$

2.1.3 Markov Decision Processes

Markov Decision Processes (MDPs) [Bel57] describe the problem of an agent acting in a dynamic world trying to maximize a reward function.
The world is defined by a finite set of states $S = \{s_0, \ldots, s_n\}$, a finite set of actions $A = \{a_0, \ldots, a_m\}$, an intitial state $s_0 \in S$, a transition model $T : S \times A \times S \to [0, 1]$ and a reward function $R : S \to [0, 1]$. The agent may specify a tactic $\pi : S \to A$ recommending an action for any given state. The quality of this tactic is defined by the expected future discounted reward over all possible trajectories of states resulting from using this tactic. The discounted future reward $U(s_{0:t})$, where t may be infinite, is defined as

$$U(s_{0:t}) = \sum_{i=0}^{t} \gamma^i R(s_i)$$

for some discount factor $\gamma \in (0, 1]$.

Solving an MDP means finding an optimal tactic π^* maximizing the expected future discounted reward. In the next section we will first generalize the described model before we return to the question of deriving such an optimal tactic.

Partially Observable Markov Decision Processes

In Partially Observable Markov Decision Processes (POMDPs) [Ast65] the state variables are hidden but the agent is given an observation $o \in O$ and an observation model $O : S \times O \to [0, 1]$. Additionally the initial state may not be deterministically known but only how it is distributed (the belief). Generally we may define the belief at any time step to be a distribution over the state space: $b_t : S \to [0, 1]$.
As the agent cannot parametrize his tactic on the state variable, he is bound to base it on his belief over it: $\pi : (S \to [0, 1]) \to A$. To be able to make a decision at any time step the agent therefore needs to update his belief based on his last belief, the action taken and the observations perceived. We know this process from section 2.1.2. It is a Bayes filter:

$$b(s_{t+1}) = P(s_{t+1}|o_{0:t+1}, a_{0:t}) = \alpha O(s_{t+1}, o_{t+1}) \sum_{s_t} T(s_t, a_t, s_{t+1}) b(s_t)$$

Now for time step $t = 0..$ the process repeats the following steps:

1. The agent chooses an action $a_t = \pi(b_t)$, where $b_t = b(S_t)$ is his current belief.
2. After executing a_t he perceives an observation o_{t+1} and updates his belief recursively according to
$$b_{t+1} = P(s_{t+1}|o_{0:t+1}, a_{0:t}) = \alpha O(s_{t+1}, o_{t+1}) \sum_{s_t} T(s_t, a_t, s_{t+1}) b_t.$$

Solving a POMDP still means finding an optimal tactic π^* maximizing the expected future discounted reward.
Alternative, "weaker", beliefs may be defined as well: E.g. the agent may map the states to different clusters representing the actions the agent takes in them. In these cases the agent will only infer a strategy optimal with respect to this approximation.
Also the state may encode it's past as well. In this case the belief may simply be a multinomial variable partitioning the state space into sets of states the agent cannot differentiate because he has not seen enough evidence to be able to. In this case the belief at time t only depends on s_t and not on b_{t-1}.

Dynamic Decision Networks

POMDPs may be defined concisely by utilizing the notational expressiveness of DBNs: Transition- and observation-model as well as the initial state distribution are already given and we add decision-, reward- and belief variables to it. A time slice of a simplified example network may be seen in figure 3.1. (a_t is a decision variable, h_t is a belief variable. In this example network, only the last time slice has reward variables. If there was one, it would depend on the state variable s_t. Generally the belief

variable depends on the belief variable in the former time slice, too.) The resulting network is called a Dynamic Decision Networks (DDN) [DW91].
In a DDN, all nodes are random variables. Therefore the decisions are modelled as non-deterministic, too. Depending on the problem to be solved, the decision variables, as well as every other variable, may still be deterministic.

2.1.4 Learning Methods

To derive an optimal strategy in this network we need to review some ideas from the domain of machine learning. The general problem is that given a Bayesian network consisting of "input" variables X and "output" variables Y, where the Y depend conditionally on X and given a set D of n independent and identically distributed data points (x_i, y_i) produced by this network we want to estimate the unknown conditional distributions in it.
This is done by assuming the conditionals of Y to be distributed according to some prior distribution depending on parameters β which are added to the network as random variables. The network takes the form seen in figure 2.1.4. Our interest lies in the distribution $P(y_0|x_0, D)$: How is y_0 distributed for

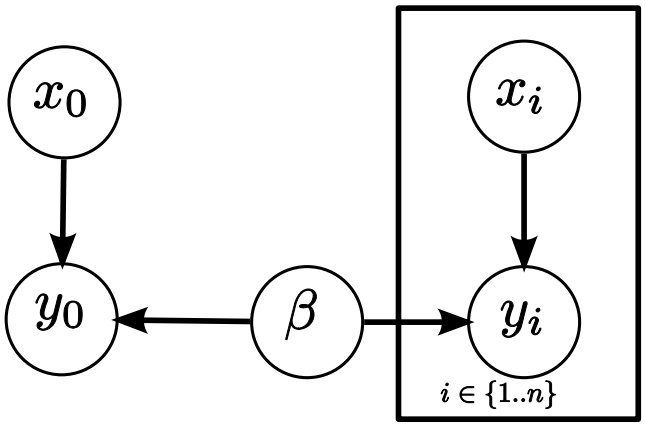

Figure 2.1: A Bayesian Network with data points (x_i, y_i), parameter β, observation x_0 and query variable y_0

a specific x_0 given that we have already seen $D = \{(x_i, y_i)\}_{i=1}^n$. Considering the underlying Bayesian network we find the following:

$$P(y_0|x_0, D) = \alpha \int_\beta P(y_0|x_0; \beta)P(D|\beta)P(\beta)d\beta$$

Likelihood Maximization

One important property of this integral is that, according to the law of large numbers, as the sample size $|D|$ approaches infinity for all β the value of the $P(D|\beta)$-term will converge faster to zero than for β^*, the one explaining the data best which will therefore completely dominate the prior and thereby

the prediction:

$$P(y_0|x_0, D) = P(y_0|x_0; \beta^*), \text{ for } \beta^* = \underset{\beta}{\operatorname{argmax}}\, P(D|\beta)$$

In some cases we are therefore content with optimizing the latter (easier one) instead. As it is called the likelihood of the data we call the process of maximizing it "Likelihood Maximization" and the resulting parameter β^* the maximum likelihood (ML) estimate [Fis22].
The process of deriving $\beta^* = \operatorname{argmax}_\beta P(D|\beta) = \operatorname{argmax}_\beta \prod_{i=1}^{n} P(y_i|x_i;\beta)$ can be simplified by first taking the logarithm of the expression:

$$\ln(P(D|\beta)) = \sum_{i=1}^{n} \ln(P(y_i|x_i;\beta))$$

The term $\ln(P(D|\beta))$ is called the log-likelihood. This operation does not change the result but among other things turns products into sums which are easier to work with when evaluating the partial derivative:

$$\frac{\partial}{\partial\beta} \sum_{i=1}^{n} \ln(P(y_i|x_i;\beta)) = 0 \Longrightarrow \beta = \beta^*$$

Expectation Maximization

When we are not able to analytically differentiate the log-likelihood because of latent (hidden) variables we can use the iterative Expectation Maximization (EM) algorithm [BP66] to approximate a local ML estimate. The general use case for the EM algorithm is when we model the Bayesian network to contain variables Z which are not represented in the data D we are planning to train our parameter β with.
The EM algorithm may be summarized as follows: Instead of maximizing the intractable

$$\ln(P(D|\beta)) = \ln(\sum_{Z} P(D, Z|\beta))$$

we instead try and maximize the lower bound

$$\ln(P(D|\beta)) - D_{KL}(q(Z)\|P(Z|D;\beta)) = \sum_{D} q(Z)\ln(P(D, Z;\beta)) + H(q).$$

On the left hand side of the equation we find that we can improve the lower bound by finding an approximation $q(Z)$ to the posterior over the latent variables $P(Z|D;\beta)$ for a fixed β while on the right hand side we see that we can improve by maximizing $\sum_D q(Z)\ln(P(D, Z;\beta))$ for a fixed $q(Z)$ with respect to β:

E-step: $q_{(t+1)} = \operatorname{argmax}_q D_{KL}(q(Z)\|P(Z|D;\beta)) = P(Z|D;\beta)$

M-step: $\beta_{(t+1)} = \operatorname{argmax}_\beta \sum_z q(z)\ln(P(D, z;\beta)) \Longleftrightarrow \frac{\partial}{\partial\beta}\sum_z q(z)\ln(P(D, z;\beta)) = 0 \Longrightarrow \beta = \beta_{t+1}$

Convergence is guaranteed as the quantity maximized is a lower bound to the actual likelihood of the ML estimate and each step we take increases it.

2.1.5 Decision-Theoretic Agents for POMDPs using Expectation Maximization

There is a host of methods for solving POMDPs. In this section we will describe an approach based on likelihood maximization [TS05]. In the case of dynamic decision networks we can utilize the EM algorithm to tune the model's parameters π so the probability of observing a high expected future discounted reward $P(R = 1; \pi)$ is maximized. Here the model's parameters are the action probabilities $\pi(a_t|b_t)$ of the agent's tactic. The "data" to train the parameters with is just the observation $R = 1$ while the state variables $S_{0:T}$, the decision variables $A_{0:T-1}$ and the belief variables $B_{0:T}$ are latent. The maximization process can be clarified as follows: While we are usually interested in tuning the parameters of a generative model to increase it's likelihood of generating some actually observed data points by tuning the parameters of the assumed underlying distributions, here we seek to maximize the model's likelihood of producing the data point "high reward" by tuning the agent's probabilities of choosing his actions, which in principle is no different: We first have to determine the posterior distribution over the latent variables

$$P(Z|D; \beta) = P(S_{0:T}, A_{0:T-1}, B_{0:T}|R = 1; \beta)$$

which is done by filtering and smoothing as described in the section on Bayes filter and then updating the agent's tactic in an M-step according to

$$\hat{\pi}(a_t|b_t) = \underset{\pi(a_t|b_t)}{\operatorname{argmax}} \sum_{s_{0:T}, a_{0:T-1}, b_{0:T}} q(s_{0:T}, a_{0:T-1}, b_{0:T}) \ln(P(R = 1, s_{0:T}, a_{0:T-1}, b_{0:T}; \beta)).$$

Cooperative POMDPs

Recent work [KZT11] demonstrated that this approach is applicable to cooperative POMDPs as well: The problem of coordinating n agents to maximize a joint reward function. The DDN has a similar form but each time slice includes n decision nodes influencing the transition from state s_t to s_{t+1}. Using the EM-schema we are able to derive update steps for each agent's strategy.

2.2 Game Theory

While the methods in the former section deal with problems concerning possibly many agents trying to maximize a single reward function we now review problems where agents try to maximize different reward functions. As these problems are generally called games the scientific branch dealing with them has been coined game theory [vNM44].

2.2.1 Normal-form Games

Normal-form Games are *one-shot simultaneous move games*: In a game with a set P of n players all $p \in P$ simultaneously choose from their respective sets of actions A_p a move $a_p \in A_p$, resulting in a joint action vector $a \in \times_{p \in P} A_p$. Determined by this joint action vector each player p receives a payoff $u_p : \times_{p \in P} A_p \to [0, 1]$. The notation $u_p(a_p, a'_{-p})$ will be useful to explain that p plays according to action vector a while the other players play according to another strategy vector a'.

Solution Concepts

Just as in the single player case we are interested in finding strategies for all agents which maximize the respective agent's expected reward. However, in contrast to the single player case, no single agent can optimize his tactic without having to consider how the other agents will adapt their strategies in reaction to this change. Therefore game theory is concerned with finding equilibria of games: Strategies no player would deviate from, even when being told the strategies of the other players.
All solution concepts we discuss in this section adhere to this idea. However each is found in a larger class of games than the one before it.

Dominant Strategy Solutions An action vector $a \in \times_{p \in P} A_p$ is a dominant strategy solution, if, considering all combinations of actions the other players might take, a_p is always p's best response: Given any alternative action vector a', p never looses by switching his action to a_p:

$$u_p(a_p, a'_{-p}) \geq u_p(a'_p, a'_{-p})$$

As each p only has to choose the one optimal action specified by the solution even telling him the actions chosen by the other players would not influence his decision.
Dominant strategy solutions are trivially found. However few games have one, which is why we need a more general concept.

Pure Nash Equilibria An action vector $a \in \times_{p \in P} A_p$ is a pure Nash equilibrium, if, given that all players except p play according to a_{-p}, a_p is p's best response. a_p is better than any alternative a'_p:

$$u_p(a_p, a_{-p}) \geq u_p(a'_p, a_{-p})$$

This concept has much less stringent requirements as it does not demand a single best response to all possible action combination of the other players but only a specific combination of actions from which no player would deviate. Games with more than one pure Nash equilibrium - there is no dominant strategy solution - are sometimes called *coordination games*: Players may communicate and agree on one of the equilibria - randomly if necessary but generally avoiding dominated solutions from which all of them would deviate to one of the equilibria.
In contrast there are games with no pure Nash equilibria: For every $a \in \times_{p \in P} A_p$ at least one p would choose an alternative a'_p when being told the actions a_{-p}.

Mixed Nash Equilibria Therefore we allow players to choose not only one specific action but a probability distribution π_p, called *mixed strategy*, over all actions. All players try to maximize their respective expected payoffs $u_p(\pi) = \sum_a u_p(a) \prod_p \pi_p(a_p)$[1], which assumes that they are risk-neutral. A *Nash equilibrium* specifies a mixed strategy π_p for every p so that given that all other players play according to π_{-p}, π_p is p's best response:

$$u_p(\pi_p, \pi_{-p}) \geq u_p(\pi'_p, \pi_{-p})$$

[1] Note the overloaded notation: u_p may denote the payoff of a joint action vector as well as the expected payoff of a joint mixed strategy to player p.

Fortunately Nash equilibria are found in all games with finite numbers of players and actions [Nas51]. It is, however, a PPAD-complete problem to find them [DGP06, GP06], which means that it is generally very hard [Pap94] (many decision problems such as "Does a Nash equilibrium that has at least some given value for one of the players exist?" are NP-complete even for two-player games [GZ89]).

Regret Minimization and Normal-form Games

Regret minimization [Han57, LS82] deals with learning in repeated single-player games: At each time step t the agent (online algorithm H) specifies a mixed strategy π^t over the available N actions $a \in A$. We observe losses $l^t : A \to [0,1]$ for all actions resulting in an expected loss $E(l^t) = \sum_a \pi^t(a) l^t(a)$. After T time steps the agent will have incurred a cumulative expected loss of $L^T = \sum_{t=0}^{T-1} E(l^t)$. The aim is to design online algorithms that, given enough training, perform as good as the best algorithm in some given simple comparison class. Such a class is usually defined by a *modification rule* $F : A \longrightarrow A$ that shifts the probability H assigned to action $a \in A$ to another action $F(a)$ thereby defining a new mixed strategy $\hat{\pi}^t(a) = \sum_{a'} I(F(a') = a)\pi^t(a')$, i.e. $\hat{\pi}^t(a)$ is the sum of the probabilities of the actions that F mapped to a. The resulting strategy incurs a cumulative expected loss of $L^T_{H.F} = \sum_t \sum_a \hat{\pi}^t(a) l^t(a)$. Generally F may or may not depend on the history of actions. Here, however, we concentrate on the memoryless case.

External Regret Minimization A simple comparison class we will need in this paper is given by the class of the N modification rules mapping all actions to the same output: $\{\forall'_a F_a(a') \mapsto a | a \in A\}$. Modifying our T decisions by F_a yields a sequence of actions $a_{0:T-1}$ which would have incurred a cumulative loss of $L^T_a = \sum_{t=0}^{T-1} l^t(a)$ [2]. The best algorithm in this comparison class minimizes this loss: $L^T_{min} = \min_a L^T_a$.
Now we can define the external regret: $R_{ex} = L^T_H - L^T_{min}$.

The Polynomial Weights Algorithm A specific external regret minimizing procedure is the polynomial weights (PW) algorithm [CBMS05]: Every action a is assigned a weight w^t_a (initially $\forall_a w^0_a = 1$) and at every time step t the probabilities assigned to the actions in the mixed strategy are proportional to their weight: $\pi^t(a) = w^t_a / \sum_{a'} w^t_{a'}$. Upon receipt of the loss l^t the algorithm updates the weights according to $w^{t+1}_a = w^t_a(1 - \eta l^t(a))$, where η is a small constant.
It turns out that the external regret of this procedure is sublinear in T:

$$L^T_{PW} \leq L^T_{min} + o(T) \Longrightarrow \lim_{T\to\infty} \frac{L^T_{PW}}{T} \leq \frac{L^T_{min}}{T}$$

In other words, given enough time, PW performs as good as always choosing the best fixed action in hindsight.

Results with Respect to Game Theory One significant result for game theory is that in a two-player repeated constant sum normal-form game a players' average reward (or loss) will approach the well-defined value the game has to the respective player, if she plays an external regret minimization algorithm with regret sublinear in T (such as PW) [CBL06].

[2] As the strategy is deterministic, this is not only an expected but a definitive loss.

While this does not guarantee that in a single game player p's reward has an expected value of at least the value of the game, as it would be if she played an equilibrium strategy, it still guarantees that for large T the average converges to the optimum.
Another comparison class named *counterfactual regret* has been described [ZJBP08], which guarantees convergence to ϵ-Nash equilibria in two-player constant sum games. We will see a coarse description of it in section 3.2.2.

2.2.2 Extensive Games

In the past different generalizations of the normal-form game model have been analysed:

Repeated Games: The players play a normal-form game a fixed or infinite number of times while being told the decisions the other agents made in the past games. One example may be repeatedly driving to work from home trying to find the best route.

Multistage Games: Actions taken result in a new state where the players may take different actions. Board games without chance elements like chess fall into this category.

Games with incomplete Information: Incomplete information may result from different conditions. Chance elements like rolling dice or shuffling card decks are one, information of one player hidden to another like private cards or the positions of the player's ships in the game *Battleship* are another.

We are primarily concerned with *extensive games*, which generalize all of them: They have the form of a tree of game states (game trees). The tree's root is the initial state, edges correspond to moves. Inner nodes are decision nodes of exactly one player - only one player chooses among the available actions - while leaves are terminal nodes labelled with the payoffs to all players.
The set of decision nodes is partitioned into sets of *information sets* $\{H^0, \dots, H^n\}$ where H^p is the set of all information sets belonging to player p. Each information set $h \in H^p$ is a set of states in which p may choose a move $a \in A_h$ but which cannot be differentiated by him/her because of incomplete information so p has to base his decision completely on the fact that he is in the information set h. Analogously, in full-information games, every information set only contains a single state. We also introduce a chance player with a known, fixed behavioural strategy to account for random events.
We define a behavioural strategy to be a distribution $\pi(A|H)$. Specifically $\Omega(H) = \bigcup_p H^p$ and $\Omega(A) = \bigcup_{p,h \in H^p} A_h$, but by definition a valid strategy of player p may only assign non-negative probabilities to $a \in A_h$ for a given $h \in H^p$.

Extensive Games and Regret Minimization

Driven by the successes of applying regret minimization algorithms to normal-form games we are interested in employing them for extensive games as well. One option is to interpret them as normal-form games where the players have to choose the moves they would take at any information set even before the game starts, i.e. each $\times_{h \in H^p} A_h$ is considered a move of player p and the algorithm needs to find the optimal probability distribution over this domain.
A more natural approach would try to find the optimal strategy $\pi(A|H)$ by considering the games at each information set individually. To make this work we would need to infer the loss vectors needed

to feed an instance of the regret minimization algorithm specifically for each information set. It turns out that in the main part of this paper we will derive a solution based on probabilistic inference.

Chapter 3

Partially Observable Markov Games

Now that we have established the background we can outline the main contribution of this paper. Markov games have been described in [Lit94]. A partially observable Markov game (POMG) is an alternative description of extensive games using the notions of POMDPs for cooperative games: For every time slice t the state variable S^t has one value for each state in the game tree - both decision nodes and terminal nodes. There is a multinomial variable H_t with one value for each $h \in \bigcup_p H^p$, all information sets of all players in the extensive game. The decision node's domain is the union of all actions available to all player in all information sets: $\bigcup_p \bigcup_{h \in H^p} A_h$. The distribution of the

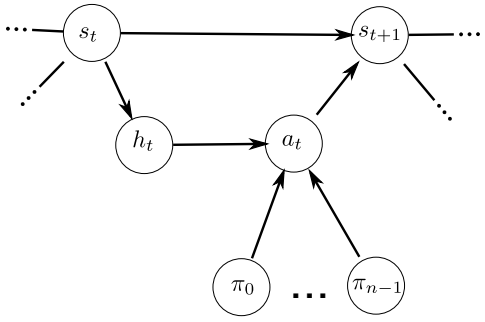

Figure 3.1: Time slice t of a POMG graph

information set variable is given by:

$$P(h_t|s_t) = \begin{cases} 1 & \text{if } s_t \in h_t, \\ 0 & \text{otherwise} \end{cases}$$

If $h_t \in H^p$ the decision node, a_t, is distributed according to $P(a_t|h_t;\pi) = \pi_p(a_t|h_t)$. Finally the transition model $P(s_t|a_{t-1}, s_{t-1})$, $t > 0$, is deterministic (1 if choosing a_{t-1} in s_{t-1} results in s_t, 0 otherwise) if there are no chance elements in the game. In games with a chance player we may ascribe his moves to the stochastic properties of the transition model: A player's move a in state s resulting in a state s' in an information set h' of the chance player, who chooses a move a' according to his static

strategy π_0, which leads to a state s'', may equivalently described as a stochastic transition model

$$P(s''|a,s) = \begin{cases} \pi_0(a'|h') & \text{if } T(s,a,s') = 1, s' \in h', h' \in H^0, T(s',a',s'') = 1 \\ 0 & \text{otherwise} \end{cases}$$

where H^0 is the set of information sets of the chance player and T is the deterministic transition model defined by the game tree of the extensive game.
Analogously we define the initial state distribution $P(s_0)$ to be deterministic (1 if s_0 is the initial state, 0 otherwise) if there is no chance player and conversely

$$P(s') = \begin{cases} \pi_0(a,h) & \text{if } s_0 \in h, h \in H^0, T(s_0,a,s') = 1 \\ 0 & \text{otherwise.} \end{cases}$$

In contrast to general POMDPs we define a finite temporal horizon T as defined by the depth of the game tree. This allows us to model the POMDP's network to only contain a single reward variable R_p for every player p conditioned on the final state s_T. If there are terminal states k at a time step $t < T$ the transition model shall specify $T(S_t = k, a, S_{t+1} = k) = 1$ for all a and consequently $T(S_t = k, a, S_{t+1} = j) = 0$ for all $j \neq k, a$: The game shall not leave the terminal state so that $s_T = k$ and the rewards observed are the ones specified for $S = k$.
As we derived the POMDP from an extensive game, we find another important property: Because, in the game tree, the states and information sets are unique to a specific time slice, by construction they are so in the POMDP, too. As we shall see in the next section, this will make deriving the strategy update simpler.

3.1 Solving POMGs: The EM Approach

One idea that comes to mind naturally is to apply the algorithms for solving cooperative POMDPs. The approach requires us to first derive an M-step for the EM-algorithm used to update the parameters (strategies) in the network.

3.1.1 Deriving the M-Step

As stated in the section on POMDPs, to apply the EM-algorithm to update the strategies we need to find a solution to the following problem:

$$\underset{\pi(a_t|b_t)}{\operatorname{argmax}} \sum_{s_{0:T}, a_{0:T-1}, b_{0:T}} P(s_{0:T}, a_{0:T-1}, b_{0:T}|R=1;\pi^{(old)}) \ln(P(s_{0:T}, a_{0:T-1}, b_{0:T}, R=1;\pi))$$

In our case the belief variable B takes the form of the information set variable H. As the terminal nodes are not part of an information set, we are only concerned with the sequence $h_{0:T-1}$. Also, we denote the respective sequences as $\underline{s}, \underline{h}$ and $\underline{a}$. Additionally we have to introduce the other players' reward variables R_{-p} as latent:

$$\forall p \in \{0..n-1\}, h \in H_p, a \in A_h :$$
$$\underset{\pi(a|h)}{\operatorname{argmax}} \sum_{\underline{s},\underline{h},\underline{a},r_{-p}} P(\underline{s}, \underline{h}, \underline{a}, r_{-p}|R_p=1;\pi^{(old)}) \ln(P(\underline{s}, \underline{h}, \underline{a}, r_{-p}, R_p=1;\pi))$$

In deriving the update for a specific $\pi(a|h)$ we shall remember that h can only be visited in a specific time step τ. Furthermore, before we can determine the partial derivative we need to introduce Lagrange multipliers to ascertain that for all information sets the action probabilities sum to 1:

$$\left(\sum_{\underline{s},\underline{h},\underline{a},r_{-p}} P(\underline{s},\underline{h},\underline{a},r_{-p}|R_p=1;\pi^{(old)})\ln(P(\underline{s},\underline{h},\underline{a},r_{-p},R_p=1;\pi))\right)-\left(\sum_h \lambda_h\left(\sum_{a_h}\pi(a_h,h)-1\right)\right)$$

We can now begin the derivation:

$$\begin{aligned}&\frac{\partial}{\partial\pi(a,h)}\left(\sum_{\underline{s},\underline{h},\underline{a},r_{-p}} P(\underline{s},\underline{h},\underline{a},r_{-p}|R_p=1;\pi^{(old)})\ln(P(\underline{s},\underline{h},\underline{a},r_{-p},R_p=1;\pi))\right)\\ -&\frac{\partial}{\partial\pi(a,h)}\left(\sum_h \lambda_h\left(\sum_{a_h}\pi(a_h,h)-1\right)\right)\end{aligned}$$

By the linearity of the derivative operator and holding $P(\underline{s},\underline{h},\underline{a},r_{-p}|R_p=1;\pi^{(old)})$ constant as required by the EM-approach we get:

$$\left(\sum_{\underline{s},\underline{h},\underline{a},r_{-p}} P(\underline{s},\underline{h},\underline{a},r_{-p}|R_p=1;\pi^{(old)})\frac{\partial}{\partial\pi(a,h)}\ln(P(\underline{s},\underline{h},\underline{a},r_{-p},R_p=1;\pi))\right)-\lambda_h$$

To continue we need to factorize $P(\underline{s},\underline{h},\underline{a},r_{-p},R_p=1;\pi)$:

$$\begin{aligned}&\left(\sum_{\underline{s},\underline{h},\underline{a},r_{-p}} P(\underline{s},\underline{h},\underline{a},r_{-p}|R_p=1;\pi^{(old)})\right.\\ &\cdot\frac{\partial}{\partial\pi(a,h)}\ln\left(P(s_0)P(r_{-p}|s_T)P(R_p=1|s_T)\left(\prod_{t=1}^{T}P(h_{t-1}|s_{t-1})\pi(a_{t-1}|h_{t-1})P(s_t|s_{t-1},a_{t-1})\right)\right)\Bigg)\\ &-\lambda_h\end{aligned}$$

Turning the logarithm of the product into a sum over the logarithms of the factors we find that $\sum_{t=1}^{T}\ln\pi(a_{t-1}|h_{t-1})$ is the only term in the resulting sum not constant with respect to the derivative, the other derivatives are therefore zero. Furthermore, since the information set h can only be visited in time step τ, only h_τ can take on the value h, so for all other t the derivative is zero, too:

$$\frac{\partial}{\partial\pi(a,h)}\sum_{t=1}^{T}\ln\pi(a_{t-1}|h_{t-1})=\frac{\partial}{\partial\pi(a,h)}\ln\pi(a_\tau|h_\tau)$$

Even then, the derivate of this term is only nonzero if $h_\tau=h$ and $a_\tau=a$:

$$\frac{\partial}{\partial\pi(a,h)}\ln\pi(a_\tau|h_\tau)=\begin{cases}\frac{1}{\pi(a|h)} & \text{if } h_\tau=h \text{ and } a_\tau=a\\ 0 & \text{otherwise.}\end{cases}$$

We express this by making use of the indicator function $I(a_\tau = a \wedge h_\tau = h)$, which is 1 if the condition is satisfied and 0 otherwise:

$$\left(\sum_{\underline{s},\underline{h},\underline{a},r_{-p}} P(\underline{s},\underline{h},\underline{a},r_{-p}|R_p=1;\pi^{(old)}) \frac{I(a_\tau = a \wedge h_\tau = h)}{\pi(a|h)} \right) - \lambda_h$$

We may simplify the sum by first specifically leaving out the summation over a_τ and h_τ as the indicator function specifies their values.

$$\frac{\sum_{\underline{s},\underline{h}_{-\tau},\underline{a}_{-\tau},r_{-p}} P(\underline{s},\underline{h}_{-\tau},h_\tau=h,\underline{a}_{-\tau},a_\tau=a,r_{-p}|R_p=1;\pi^{(old)})}{\pi(a|h)} - \lambda_h$$

Recognizing a marginal distribution, we may simplify again:

$$\frac{P(h_\tau=h,a_\tau=a|R_p=1;\pi^{(old)})}{\pi(a|h)} - \lambda_h \tag{3.1}$$

All that's left is to equate all partial derivatives to zero and solve the resulting set of equations. First we solve (3.1) for $\pi(a|h)$:

$$\begin{aligned} 0 &= \frac{P(h_\tau=h,a_\tau=a|R_p=1;\pi^{(old)})}{\pi(a|h)} - \lambda_h \\ \iff \pi(a|h) &= \frac{P(h_\tau=h,a_\tau=a|R_p=1;\pi^{(old)})}{\lambda_h} \end{aligned} \tag{3.2}$$

By the constraint $\sum_a \pi(a|h) = 1$ follows:

$$\begin{aligned} \iff \lambda_h &= \sum_a P(h_\tau=h,a_\tau=a|R_p=1;\pi^{(old)}) \\ \iff \lambda_h &= P(h_\tau=h|R_p=1;\pi^{(old)}) \end{aligned}$$

Substituting back into (3.2) we arrive at an interesting result:

$$\begin{aligned} \pi(a|h) &= \frac{P(h_\tau=h,a_\tau=a|R_p=1;\pi^{(old)})}{\lambda_h} \\ &= P(a_\tau=a|h_\tau=h,R_p=1;\pi^{(old)}) \end{aligned}$$

This update rule specifies that player p can increase her likelihood of observing $R_p = 1$ by setting her action probabilities to the frequencies of playing the respective action in the cases she won!
As this quantity is not readily available we need to reduce it to familiar expressions.

$$\begin{aligned} \pi(a|h) &= P(a_\tau=a|h_\tau=h,R_p=1;\pi^{(old)}) \\ &\propto P(a_\tau=a,h_\tau=h,R_p=1;\pi^{(old)}) \\ &= \sum_{s_\tau,s_{\tau+1}} P(s_\tau,h_\tau=h,a_\tau=a,s_{\tau+1},R_p=1;\pi^{(old)}) \\ &= \sum_{s_\tau,s_{\tau+1}} P(R_p=1|s_{\tau+1};\pi^{(old)})P(s_{\tau+1}|a_\tau=a,s_\tau)\pi^{(old)}(a,h)P(h_\tau=h|s_\tau)P(s_\tau;\pi^{(old)}) \\ &= \pi^{(old)}(a,h) \sum_{s_\tau,s_{\tau+1}} \beta^p_{\tau+1}(s_{\tau+1})P(s_{\tau+1}|a_\tau=a,s_\tau)P(h_\tau=h|s_\tau)\alpha(s_\tau) \end{aligned}$$

The factors $P(s_{\tau+1}|a_\tau = a, s_\tau)$, $\pi^{(old)}(a, h)$ and $P(h_\tau = h|s_\tau)$ are given by the transition model, the current strategy and the partition of states into information sets, respectively. $\alpha(s_\tau)$ and $\beta^p_{\tau+1}(s_{\tau+1})$ are the results of filtering: The first is the forward message (prediction), the latter is the backward message for player p (using reward variable R_p).

3.1.2 The Algorithm

Having derived the M-step we can now formulate the whole algorithm used in this approach. A framework automating Bayesian inference makes it trivial to implement it as it consists of nothing else. In algorithms (1) and (2), however, we describe a more basic implementation based on the forward-backward algorithm for HMMs.

Algorithm 1 Training Step

for all $\{(p, a, h, s, s')|p \in \{0..n-1\}, h \in H_p, a \in A_h, s \in h, s' \in S, T(s'|a, s) > 0\}$ **do**
  $T'(s, s') \leftarrow T(s'|s, a)\pi_p(a, h)$
end for
for all $s \in S$ **do**
  $\alpha_0(s) \leftarrow P(S_0 = s)$
end for
for all $\{(t, s', s)|t \in \{1..T\}, s' \in S, s \in S\}$ **do**
  $\alpha_t(s') \leftarrow \alpha_t(s') + T'(s, s')\alpha_{t-1}(s)$ // forward messages
end for
for $p = 0 \rightarrow n$ **do**
  for all $s \in S$ **do**
    $\beta'(s) \leftarrow P(R_p = 1|S_T = s)$
  end for
  for $t = T - 1 \rightarrow 0$ **do**
    for all $(s, s') \in S \times S$ **do**
      $\beta(s) \leftarrow \beta(s) + T'(s', s)\beta'(s')$ // backward messages
    end for
    for all $\{(h, a, s, s')|h \in H_p, a \in A_h, s \in h, s' \in S, T(s'|a, s) > 0\}$ **do**
      $r_p(h, a) \leftarrow r_p(h, a) + \beta'(s')T(s'|a, s)\alpha_t(s)$
    end for
    for all $s \in S$ **do**
      $\beta'(s) \leftarrow \beta(s)$
    end for
  end for
end for
exact_M_Step(r)

Algorithm 2 *exact_M_Step*(r)

for all $\{(a,h)|p \in \{0..n-1\}, h \in H_p, a \in A_h\}$ **do**
  $\hat{\pi}_p(a,h) \leftarrow \pi_p(a,h) r_p(h,a)$
end for
$normalize(\hat{\pi})$

3.1.3 Results

To evaluate the algorithm, we applied it to a number of small extensive form games with known equilibria. While we present them as normal-form games, we arrived at our results by implementing them as extensive games.

The Prisoner's Dilemma Two alleged delinquents both have the options of confessing or keeping silent which result in the following losses (sentences) for the players:

	confess	keep silent
confess	(4,4)	(1,5)
keep silent	(5,1)	(2,2)

The first and second values are the losses to the row and the column player, respectively.
In this game (confess,confess) is a dominant strategy solution: Whichever choice the respective other player makes, confessing always results in a smaller loss. As we can see in figure 3.2, the algorithm converges to it after at most 20 updates (faster if we initialize the strategy priors closer to the equilibrium).

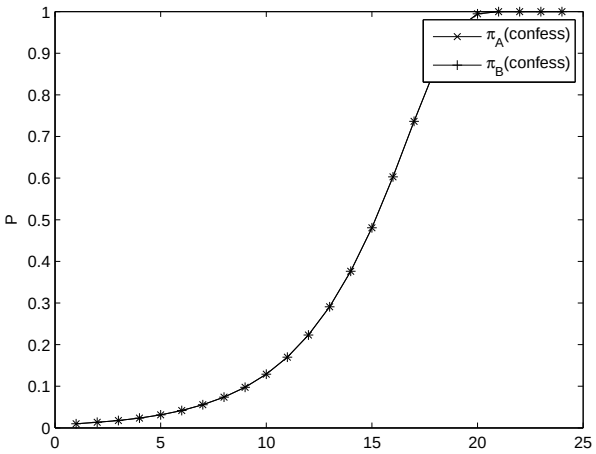

Figure 3.2: Convergence of the exact M-step algorithm to the dominant strategy solution of the prisoner's dilemma

The Battle of the Sexes In this game two players both derive higher payoffs from choosing the same moves:

	A	B
A	(6,5)	(2,2)
B	(1,1)	(5,6)

We denote the row player as A and the column player as B since they prefer the respective outcomes. Here we have two pure Nash equilibria, (A,A) and (B,B), as well as a mixed Nash equilibrium where A plays $\langle A : 0.625, B : 0.375\rangle$, B plays $\langle A : 0.375, B : 0.625\rangle$ and both players' expected value is 3.5. Figure 3.3 shows what happens, when we set the strategy priors close to one of the unstable strategy profiles (A,B) or (B,A) (one player chooses A with a probability of at least 0.5 while the other chooses B with a probability of at least 0.5): The algorithm converges to the mixed Nash equilibrium.

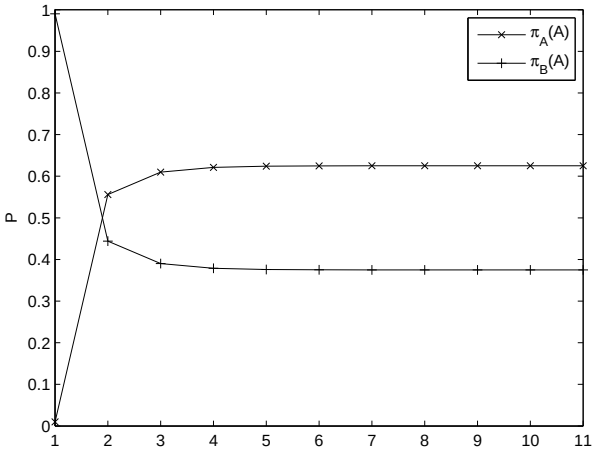

Figure 3.3: Convergence of the exact M-step algorithm to the mixed Nash equilibrium of the Battle of the Sexes

In 3.4, however, we initialized the strategy priors closer to one of the pure equilibria (both players assign a probability of at least 0.5 to the same choice) and found that the algorithm converged there.

Two Finger Morra In this zero sum game the two players both simultaneously choose to show either one or two fingers. One is the "evens" and the other one the "odds" player, where the first one wins an amount proportional to the total number of fingers from the other player if it is even and vice versa if it is odd.

	one	two
one	(2,-2)	(-3,3)
two	(-3,3)	(4,-4)

In the pairs in table 3.1.3 the first value is the payoff to the "evens" player while the second ones are the payoffs to the "odds" player.
As this game is a two-player zero sum game, we know that it has no pure Nash equilibrium and exactly one mixed Nash equilibrium [Owe82], which may be found via linear programming: Both players should

choose "one" with probability $\frac{7}{12}$ and accordingly "two" with probability $\frac{5}{12}$. The resulting values are $-\frac{1}{12}$ for the "evens" player and analogously $\frac{1}{12}$ for the "odds" player. Transforming the values to the interval $[0, 1]$ ($f(v) = (v + 4)/8$), the values correspond to ~ 0.4896 and ~ 0.5104, respectively.
Interestingly, the algorithm did not reach convergence in this game (see figure 3.5). Even more interesting, we found that the average payoffs oscillated around the expected payoffs seemingly *underdamped* (figure 3.6). These oscillations are somewhat similar to the oscilatory effects in the predator-prey model [Lot25, Vol28] In itself this could be regarded as a positive result (the values do converge!). However, another problem occurred: During the oscillation the action probabilities occasionally become so small as to cause arithmetic underflow after about 4000 iterations.

3.2 Other Approaches

The results seem to imply that the exact M-step finds pure and mixed equilibria, given that there are pure Nash equilibria. In the case of games without pure Nash equilibria, however, the learning process does not converge but the average payoff would, if arithmetic underflow did not occur. Let us therefore take a step back and analyse differences between our approach and ones we found elsewhere.

3.2.1 The External Regret Approach

Drawing on the sections on game theory and regret minimization we may remember that two players using an online algorithm with sublinear external regret to play a constant sum game guarantees that their average payoffs approach those given by the value of the game, which corresponds to our observations in the application of the exact M-step to Morra.
As an example for external regret minimization procedures we learned about the Polynomial Weights algorithm. In hindsight we recognize that our approach already implements most of the machinery necessary to implement it's update rule: We are already updating our action probabilities by multiplying them with the expected rewards and renormalizing. To equate this rule and the Polynomial Weights algorithm we simply have to update the strategy according to the cumulative moving average over all strategies seen so far. These smoothed updates are similar to those used in stochastic EM providing convergence guarantees [DLM99].
An implementation is given in algorithm (3).

Results

Replacing *exact_M_Step* by *externalRegretUpdate* the average rewards converge even in Morra (figure 3.7).

3.2.2 The Counterfactual Regret Approach

As the external regret minimization approach still did not converge to an equilibrium in strategies we then compared it to the counterfactual regret update rule. Again we found that barely any change was necessary to incorporate it into our algorithm. In the training step procedure we have to change the update of the r-vector (line 19 of algorithm (1)) to

$$r(h, a) \leftarrow r(h, a) + (\beta'(s') - \beta(s))T(s'|a, s)\alpha_t(s).$$

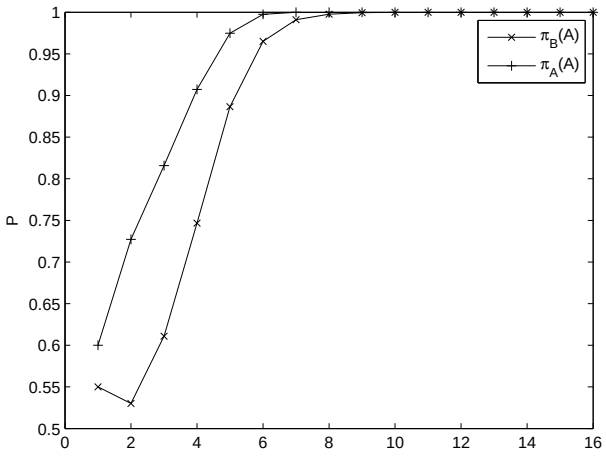

Figure 3.4: Convergence of the exact M-step algorithm to a pure Nash equilibrium of the Battle of the Sexes

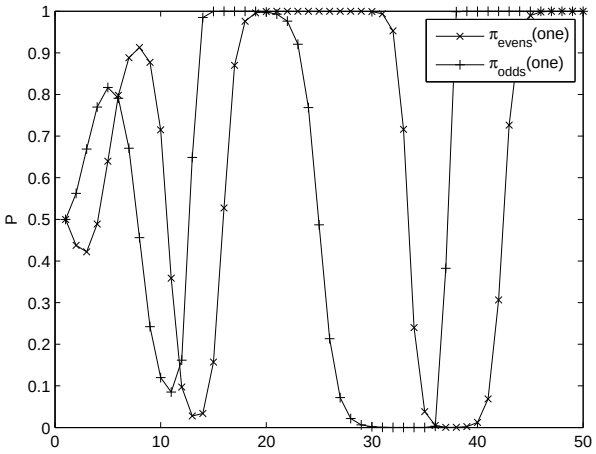

Figure 3.5: Divergent behaviour of the exact M-step algorithm in the game Morra

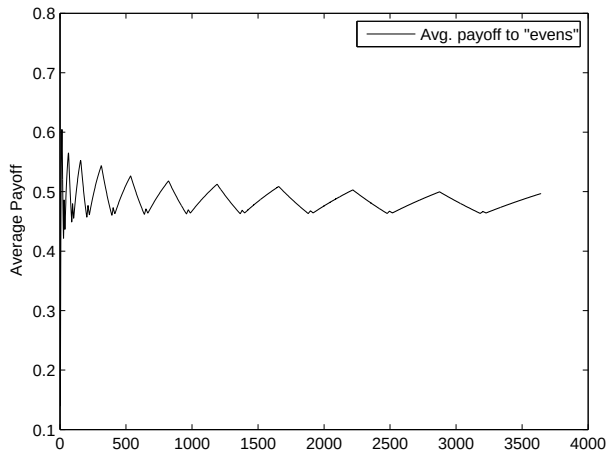

Figure 3.6: Underdamped oscillation of the average payoff in the game Morra (exact M-step)

Algorithm 3 *externalRegretUpdate*(r)

$i \leftarrow i + 1$ // A static variable incremented on every strategy update
$c \leftarrow 1/i$;
for all $\{h | p \in \{0..n-1\}, h \in H_p\}$ **do**
  $sum \leftarrow 0$
  for all $a \in A_h$ **do**
    $\hat{\pi}(a,h) \leftarrow \pi(a,h)(1 + c \cdot r(h,a))$
    $sum \leftarrow sum + \hat{\pi}(a,h)$
  end for
  for all $a \in A_h$ **do**
    $\hat{\pi}(a,h) \leftarrow \hat{\pi}(a,h)/sum$
  end for
end for
$normalize(\hat{\pi})$

So instead of basing our update on the expected value of the action, we will base it on the expected regret to the average value of all available actions. The update procedure has to be modified accordingly, as given in algorithm (4).

Algorithm 4 $counterFactualRegretUpdate(r)$

$i \leftarrow i + 1$
$c \leftarrow 1/i$;
for all $\{h | p \in \{0..n-1\}, h \in H_p\}$ **do**
  $sum \leftarrow 0$
  for all $a \in A_h$ **do**
    $r(h,a) \leftarrow \max(0, r(h,a))$
    $sum \leftarrow sum + r(h,a)$
  end for
  if $sum > 0$ **then**
    for all $a \in A_h$ **do**
      $r(h,a) \leftarrow r(h,a)/sum$
    end for
  else
    $r(h,a) \leftarrow \pi(a,h)$
  end if
  for all $a \in A_h$ **do**
    $\hat{\pi}(a,h) \leftarrow \pi(a,h)(1 + c \cdot r(h,a))$
  end for
end for
$normalize(\hat{\pi})$

Results

Modifying the update of r and replacing $externalRegretUpdate$ by $counterFactualRegretUpdate$ achieves the results we were hoping for (figure 3.8).

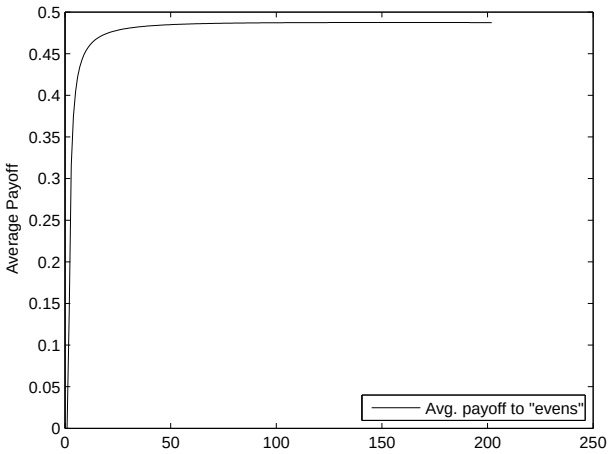

Figure 3.7: Convergence of the average payoff in the game Morra (PW algorithm)

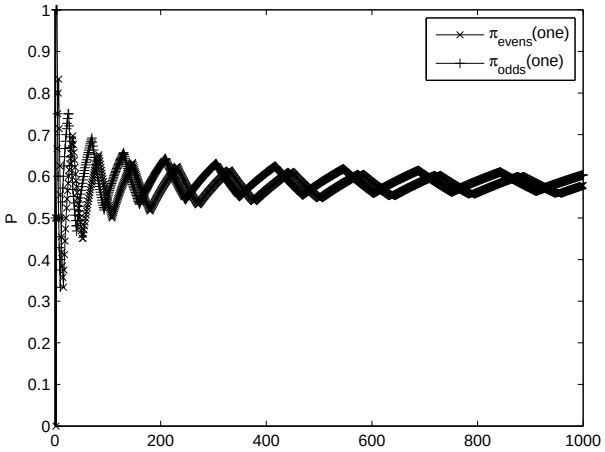

Figure 3.8: Convergence of the strategies in the game Morra (Counterfactual regret algorithm)

Chapter 4

Conclusion

Applying the derived EM algorithm to a number of games our results suggest that strategies converged to Nash equilibria if at least one pure Nash equilibrium exists and replacing the M-step update routine of our algorithm by regret minimization procedures yields even stronger results. We thereby found that our algorithm is general enough to be used as a framework for building agents with different strategy update rules for different extensive games.
Some observations we consider noteworthy are the following: The derived EM algorithm seems to be an underdamped version of the PW algorithm with similar convergence guarantees. It might be informative to analyse it's external regret. Actually we found remarkable connections between the likelihood maximization reinforcement learning approach and the regret minimization methods that suggest that generally the application of reinforcement learning methods to games may be analysed in this light. As we mentioned in the introduction, reinforcement learning researchers should be interested in regret guarantees for their algorithms. As we elucidated the similarities between regret minimization and reinforcement learning algorithms and even found evidence that supports hypotheses for regret bounds to these algorithms, we suppose that this warrants further attention.
We did not implement and test the swap regret minimization update routine with our algorithm. We expect it to show the same convergence properties as in the repeated normal-form case, but have no empirical evidence to support this hypothesis. One interesting aspect of the specific algorithm given in [BM05] is that it involves finding a stable distribution of a Markov process, so we would describe a Markov process inside of a Markov process, which could suggest an even closer relationship between the two ideas and might allow simplification.
Further research should probably concentrate on the integration of approximate inference methods and factorized state representations. The first also means to tackle the partial information problem: Using approximate inference techniques often implies sampling the problem space, so in one training step we would not get to see the rewards for *all* action sequences but only for a single one. However, this problem has also already been dealt with extensively in the regret theory domain (e.g. [CBLS06]) so we would merely have to analyse the methods provided in the context of our algorithm. Factorized state representations would allow us to describe the partial information aspects of games more naturally, in terms of state variables that are observable by one agent but hidden to the other, so players would actually have to calculate their beliefs. Also, they allow sparser transition models and therefore better performance. It might also allow dimensionality reduction methods to be applied. Combining this idea

with approximate inference techniques might yield an agent design capable of learning approximately optimal strategies based on properties of states and might therefore be able to generalize without having experts handcraft heuristics for every game to be solved and still benefit from the performance guarantees given by the regret minimization elements of the algorithm.

Bibliography

[Ast65] K.J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[Att03] H. Attias. Planning by probabilistic inference. In *Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics*. Citeseer, 2003.

[Aum74] Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

[Bel57] Richard Bellman. A Markovian Decision Process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[BM05] Avrim Blum and Yishay Mansour. From external to internal regret. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin / Heidelberg, 2005.

[BP66] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[CBL06] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge Univ Press, 2006.

[CBLS06] N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Regret minimization under partial monitoring. In *Information Theory Workshop, 2006. ITW'06 Punta del Este. IEEE*, pages 72–76. IEEE, 2006.

[CBMS05] Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 57–91. Springer Berlin / Heidelberg, 2005.

[DGP06] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 71–78. ACM, 2006.

[DK89] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

[DLM99] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the em algorithm. *Annals of Statistics*, pages 94–128, 1999.

[Doh09] Lisa Dohrmann. Latex thesis template. http://www.inf.fu-berlin.de/wiki/bin/viewfile/SE/ThesisRules?rev=1;filename=thesis_latex_template.zip, 2009.

[DW91] T. Dean and M. P. Wellman. Planning and Control. In *Temporally Flexible Inference*, chapter 8.3, pages 353–363. Morgan Kaufmann, 1991.

[EL03] Ehud and Lehrer. A wide range no-regret theorem. *Games and Economic Behavior*, 42(1):101 – 115, 2003.

[Fis22] R.A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309, 1922.

[Gho04] I. Ghory. Reinforcement learning in board games. *Technical Report CSTR-04-004, Department of Computer Science, University of Bristol*, 2004.

[GP06] P.W. Goldberg and C.H. Papadimitriou. Reducibility among equilibrium problems. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 61–70. ACM, 2006.

[GZ89] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1(1):80–93, 1989.

[Han57] J. Hannan. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[Joh07] M.B. Johanson. Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. In *Masters Abstracts International*, volume 46, 2007.

[KZT11] Akshat Kumar, Shlomo Zilberstein, and Marc Toussaint. Scalable Multiagent Planning Using Probabilistic Inference. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 3, pages 2140–2146, 2011.

[Lit94] M.L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157163. Citeseer, 1994.

[Lot25] A.J. Lotka. *Elements of physical biology*. Williams & Wilkins company, 1925.

[LPK] A. W. Moore L. P. Kaelbling, M. L. Littman. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*.

[LS82] G. Loomes and R. Sugden. Regret theory: An alternative theory of rational choice under uncertainty. *The Economic Journal*, 92(368):805–824, 1982.

[Mar13] A. A. Markov. An example of statistical investigation in the text of "Eugene Onegin" illustrating coupling of "tests" in chains. In *Proceedings of the Academy of Sciences of St. Petersburg*, volume 7, 1913.

[Nas51] John Forbes Nash. Non-cooperative Games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[Owe82] G. Owen. *Game Theory*. Academic Press, 1982.

[Pap94] C.H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[Pea85] J. Pearl. Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. In *Proceedings of the seventh Conference of the Cognitive Science Society*, pages 329–334, 1985.

[RN04] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2004.

[Sam59] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[Sam67] Arthur L. Samuel. Some studies in machine learning using the game of checkers II - Recent progress. *IBM Journal of research and development*, 11(6):601–617, 1967.

[SD69] J.R. Slagle and J.E. Dixon. Experiments with some programs that search game trees. *Journal of the ACM (JACM)*, 16(2):189–207, 1969.

[Tes92] Gerald Tesauro. Practical issues in temporal difference learning. *Machine learning*, 8(3):257–277, 1992.

[Tes95] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[TS05] Marc Toussaint and Amos Storkey. Probabilistic inference for computing optimal policies in MDPs. NIPS 2005 Workshop - Game Theory, Machine Learning and Reasoning under Uncertainty, 2005.

[vNM44] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[Vol28] V. Volterra. Variations and fluctuations of the number of individuals in animal species living together. *Journal du Conseil*, 3(1):3, 1928.

[WG07] Y. Wang and S. Gelly. Modifications of UCT and sequence-like simulations for Monte-Carlo Go. In *IEEE Symposium on Computational Intelligence and Games, Honolulu, Hawaii*, pages 175–182, 2007.

[Zer12] Ernst Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the Fifth International Congress of Mathematicians, Cambridge*, volume 2, pages 501–10, 1912.

[ZJBP08] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. *Advances in Neural Information Processing Systems*, 20:1729–1736, 2008.