EEE 448/548 - Project Report

Markov games as a framework for multi-agent reinforcement learning

Onurcan Ataç

1 Introduction

The paper "Markov games as a framework for multi-agent reinforcement learning" by Michael L. Littman explores Markov games as a framework for multi-agent reinforcement learning [1]. The paper points to how the secondary agents can only be a part of the environment as fixed in their actions in the solipsistic view of the Markov Decision Processes. In a more realistic scenario, the agent has to interact with other agents in the environment. However, the assumption of a stationary environment in the theory of Markov Decision Processes prevents including multiple agents.

Therefore, the paper suggests using the framework of Markov games to include multiple adaptive agents with interacting or competing goals in the same environment. However, in this paper, this idea is implemented only with two agents that have diametrically opposed goals.

This report will aim to analyze the theoretical findings of the specified paper, along with an attempt at a practical algorithm. The report will analyze the differences between the definitions of MDPs and Markov Games, and examine the differences between the definitions of optimal policies in terms of MDPs and Markov games. The methods to find these optimal policies for matrix games, MDPs, and Markov games will be compared, minimax-Q algorithm for a specific case for zero-sum two-player games will be introduced.

Afterwards, the soccer experiment that is implemented in the paper will be explained in detail, its results and the causes of these results will be investigated. An attempt of a similar game will be described, and the report will end with a discussion of the paper.

2 MDP and Markov Game Principles

An MDP is defined by a set of states S, and and available actions for the agent A. A transition function denotes how the state of the environment changes in response to the actions of the agent (1), and the reward function denotes the reward that the agent receives by transitioning to a state with an action (2).

$$T : S \times A \longrightarrow \mathbb{P}(S) \tag{1}$$

$$R : S \times A \times S' \longrightarrow \mathbb{R} \tag{2}$$

A Markov game also contains set of states S in its definition. Differently from an MDP, a collection of action sets is defined as $A_1, \ldots, A_k$ since there are multiple agents in the environment (one action set for each of them). As expected, every action set contributes to the transition

function (3). Each agent has its associated reward function as well (4). Each reward function seeks to maximize the expected sum of discounted rewards of its associated agent.

$$T : S \times A_1 \times \ldots \times A_k \to \text{PD}(S). \tag{3}$$

$$R_i : S \times A_1 \times \ldots \times A_k \to \mathbb{R}. \tag{4}$$

The paper adopts a specialization named two-player zero-sum Markov games, where there are only two agents and they have diametrically opposed goals. This selection is made in order to ensure a single reward function usage that one agent attempts to maximize and the opponent agent attempts to minimize, *R(s, a, o)*. In this function, *a* denotes the action that the agent takes, *s* denotes the current state and *o* denotes the action that the opponent agent takes. Consequently, cooperation cannot be considered in the paper.

3 Optimal Policy Properties

The definition of an optimal policy for MDPs is a policy that maximizes the value function. For a policy to be declared as optimal, there should not exist any other state from which any other policy can achieve a larger sum for the value function. This property is referred to as being *undominated* in the paper.

For any MDP, there exists at least one optimal policy, and at least one of those policies is stationary and deterministic. However, an optimal policy does not exist for most of the Markov games, because there cannot exist an *undominated* policy if the choice of the opponent can change the performance of the policy drastically.

In response to this problem, the paper decides to determine and use a performance measure in order to determine an "optimal" policy. According to this measure, each policy is measured by its performance against its best opponent (worst performance of the policy). Consequently, this measure selects relatively more conservative policies since it attempts to maximize the worst performance.

This new definition ensures every Markov game has at least one optimal stationary policy. However, there need not be a deterministic policy. The optimal policy can also be probabilistic.

4 Methods to Find Optimal Policies

In order to derive a method to find the optimal policy of a Markov game, the methods to find optimal policies for matrix games and MDPs should be retraced.

A matrix game can be defined as a table that has immediate rewards R_i, j for the agent that chooses action *i* where the opponent chooses the action *j*. The agent aims to maximize its expected reward, the opponent aims to minimize the expected reward of the agent. The policy of the agent becomes a probability distribution over possible actions.

According to the optimality definition done in the previous section, that is to try and find the largest value of the minimum expected reward, identifying the largest V value where there is some value of π that makes the constraints hold (5).

$$V = \max_{\pi \in \text{PD}(A)} \min_{o \in O} \sum_{a \in A} R_{o,a} \cdot \pi_a \tag{5}$$

For MDPs, a method to find the optimal policy is value iteration. The value iteration method utilizes the V and Q functions and iteratively estimates new values for V and Q (6)(7).

The reward can be maximized since in the case of the Q function, the greedy strategy leads to the optimal solution. If the best action for each state is selected, the reward can be maximized.

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s') \tag{6}$$

$$V(s) = \max_{a' \in A} Q(s,a') \tag{7}$$

If V(s) is defined as the expected reward for the optimal policy from state s, and Q(s, a, o) as the expected reward when taking the action a, at state s, as the opponent takes the action o continuing optimally (8)(9).

$$V(s) = \max_{\pi \in \text{PD}(A)} \min_{o \in O} \sum_{a \in A} Q(s,a,o)\pi_a \tag{8}$$

$$Q(s,a,o) = R(s,a,o) + \gamma \sum_{s' \in S} T(s,a,o,s')V(s') \tag{9}$$

Two equations that are derived for Markov games are very similar to equations (6) and (7) with respect to their occupation.

5 Learning Optimal Policies, Q-Learning and Minimax-Q

Besides the traditional way of solving MDPs with value iteration, the paper presents the model-free Q-learning method by Watkins. Since it is model-free, it does not utilize the transition function *T* as well.

Update function for Q-learning is as in (10). The probability of this update taking place enables us to not use *T(s, a, s')*. Q-learning converges to the true V and Q as the weighted average of the old and new estimates is taken indefinitely.

$$Q(s,a) = r + \gamma \cdot V(s') \tag{10}$$

The paper then introduces the minimax-Q algorithm, which is a blend of Q-learning and linear programming. Linear programming utilizes linear programming in order to reach the minimax (optimality measure mentioned in sections 3,4) instead of the max.

6 The Soccer Experiment

The paper consists of a soccer game implementation, in which the minimax-Q algorithm is used to train a learner. The game is played on a 4x5 grid. There are 2 players A and B. There is a ball that one of the players possesses. The moves of the agents are selected, and then they are executed in random order. Each agent can choose from 5 actions: North, South, East, West, and Stand.

For the possession change to happen, a player should execute an action that would take the player to the square occupied by the other player. The possession of the ball goes to the other player in this situation, and the move does not take place. This makes standing where the other player wants to go a good defensive choice. The possession is given to one or the other player in the beginning randomly.

When the player with the ball steps into the opposing goal, the player scores a point and players are placed into their initial configuration.

7 Training The Soccer Experiment

In the paper, the training is done over learning four different policies. The minimax-Q algorithm and Q-learning algorithm are used for policy learning. Both learners (minimax-Q learner and Q-learning learner) were trained against a random opponent and against another learner with the same algorithm as they use. Resulting policies are named MR, MM, QR, and QQ.

The pseudocode for the used minimax-Q algorithm is given in the paper. According to the paper, the Q-learning algorithm used is identical to the minimax-Q algorithm given, except max operator instead of minimax. Moreover, the Q-table does not keep the information of the opponent's action, unlike the minimax-Q algorithm.

Important Note: For MM and QQ, which are trained against another identical learner, the Q and V-tables were separated.

8 Evaluation of the Resulting Policies

The resulting policies from the experiment are evaluated in three ways:

- Competing against a random policy.
- Competing against a hand-built deterministic policy.
- Competing against a "challenger" opponent, different for each policy.

The minimax-Q learner that is trained against a random policy (MR) won 99.3% of the 6500 games against the random policy, 48.1% of the 4300 games against the hand-built deterministic policy, and 35.0% of the 4300 games against its "challenger" policy.

The minimax-Q learner that is trained against its identical version (MM) won 99.3% of the 7200 games against the random policy, 53.7% of the 5300 games against the hand-built deterministic policy, and 37.5% of the 4400 games against its "challenger" policy.

The Q-learning agent that is trained against a random policy (QR) won 99.4% of the 11300 games against the random policy, 26.1% of the 14300 games against the hand-built deterministic policy, and 0.0% of the 5500 games against its "challenger" policy.

The Q-learning agent that is trained against its identical version (QQ) won 99.5% of the 8600 games against the random policy, 76.3% of the 3300 games against the hand-built deterministic policy, and 0.0% of the 1200 games against its "challenger" policy.

Note: The total games are the number of games that can be played in 100,000 steps for each evaluation.

Competing with the random opponent, every policy performed very well. However, QR performed better than MR by a small margin. This might be because MR considers the actions against an ”idealized opponent”, which causes the maximum loss for MR. Therefore, it does not directly train to beat this opponent.

Competing with the hand-built deterministic opponent, MM, and MR did significantly better than QR but significantly worse than QQ. This shows both the possible positive and negative effects of using minimax rather than max since it is more conservative and maximizes the minimum performance against the opponent. By doing that, minimax-Q raises the base but brings the ceiling down. QQ implemented a way better defense and QR implemented a worse defense by chance. Any change in the opponent can create massive swings in the performances of QQ and QR.

Even though there is a minimal difference between the performances of MM and MR on the hand-built deterministic opponent, there being a difference shows that the optimal policy is not found yet against the opponent.

Competing with the ”challenger” is where MR and MM models excelled as expected. This is because the ”challenger” policies were trained to expose and bring out the worst-case performances from the policies. As expected from minimax-Q, the worst-case winning rates of MR and MM came out significantly higher than QR and QQ.

9 My Attempt: The Soccer Experiment

In order to make an experiment such as the soccer experiment in the paper myself, I attempted to code a different soccer experiment myself. I managed to set a 4x5 grid and two spots in the grid that symbolize the goals of the two agents.

I then set the two agents and the rules likewise to the experiment in the paper. I was able to simulate two agents moving in the ways that the agents move in the soccer experiment in the paper. Action execution orders are selected randomly. Letters L and R symbolize the goals of the agents A and B. This can be seen in the Figure 1.

I also implemented the minimax-Q learner agent and the random agent, along with a bunch of other agents. However, when I run the code for a game between a minimax-Q agent and a random agent, I get a straight learning curve plot for the minimax-Q agent when its win percentage over time is printed.

Therefore, I determined that the part of my code which is about the minimax-Q agent is faulty.

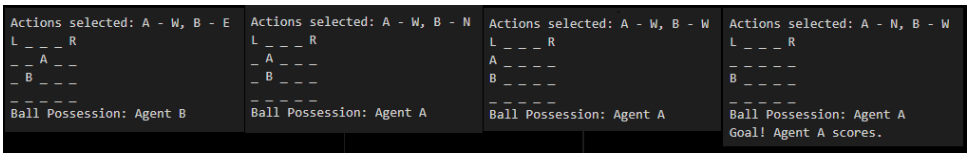

Figure 1: Example sequence of the experiment.

However, I could not determine the cause of the fault, and therefore my attempt at the soccer experiment did not work correctly.

10 Discussion of the Paper

The paper presented a valuable contribution to the literature, by introducing valuable ideas about the Markov games and simulating multiple agents in environments. The ideas such as the minimax-Q algorithm that stem from this paper are used in other research papers such as the ”Friend-or-Foe Q-learning in General-Sum Games” as well [2].

The idea of minimax-Q algorithm and the minimax-Q agent is also valuable in terms of creating a conservative algorithm that will not fall to the potential tricks of an interacting agent that aims to develop an inaccurate policy in its opponent agent.

11 References

[1] M. L. Littman, “Markov games as a framework for multi-agent Reinforcement Learning,” Machine Learning Proceedings 1994, pp. 157–163, 1994. doi:10.1016/b978-1-55860-335-6.50027-1

[2] M. L. Littman, ”Friend-or-foe Q-learning in general-sum games,” in ICML, vol. 1, pp. 322-328, June 2001.