MULTI-AGENT REINFORCEMENT LEARNING AND TWO PLAYERS ZERO-SUM STOCHASTIC GAMES

Yiqi Jiang
yj89@cornell.edu

Haiyang Yu
hy397@cornell.edu

December 12, 2021

ABSTRACT

In this project, we investigate value function approximation in the context of two players zero-sum Markov games. We start with the Markov decision process (MDP), a setting for single agent interacting with an environment defined by a probabilistic transition function. In this particular setting, we analyzes two algorithms, value iteration and Q-learning. We then proceed to the framework of Markov games for multiple adaptive agents' setting, focusing specifically on zero-sum games. The project reviews the minimax value iteration and the minimax Q-learning algorithms designed for two players zero-sum Markov games. We demonstrate the viability of the two algorithms by designing two Markov games and calculating the corresponding Nash equilibrium values.

***K*eywords** MARL · MDP · Markov Games · Zero Sum Game

1 INTRODUCTION

Reinforcement learning (RL) is a subfield of machine learning (ML), where agents learn how to behave optimally based on a trial-and-error procedure during their interaction with the environment. Unlike supervised learning, the actions taken by an agent in RL have consequences, and the agent need to make sequence of decisions to complete a task. An RL agent attempts to find the optimal policy that maximise its long-term reward. The process of finding such optimal policy is formulated by MDPs.

Following the remarkable success of the Alpha Go presented in 2019, multi-agent reinforcement learning (MARL) receives significant advances. It is an interdisciplinary domain that includes RL, game theory, optimization, and stochastic controls. The MARL setting can be placed into three categories, fully cooperative, fully competitive, and a mixed of two. It can also be classified into two categories, fully observable and partially observable. In the cooperative setting, agents collaborate to optimize a common long-term return; while in the competitive setting, the return of agents usually sum up to zero. The mixed setting involves both cooperative and competitive agents, with general-sum returns.

Markov games can be viewed as generalizations of classical game theory and MDP framework. In this project, we focus on two players zero-sum stochastic games, a fully observable and fully competitive game, in which two players make simultaneous decisions in the same environment with shared state information. The reward function and the state transition probabilities depend

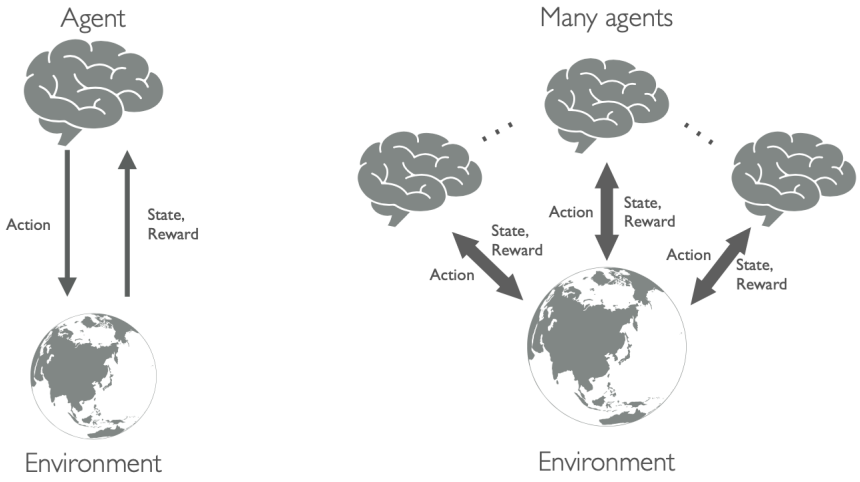


Figure 1: Diagram of a single-agent MDP (left) and a multi-agent Markov game (right). [3]

on the current state and the current agents' joint actions. The reward function in each state is the payoff matrix of a zero-sum game. In such a case, agents need to take into account and interact with not only the environment but also other learning agents. A decision-making process that involves multiple agents is usually modelled through a stochastic game[1], also known as a Markov game[2].

2 DEFINITIONS

In this section, we provide the necessary background on reinforcement learning, i.e., MDPs for single-agent reinforcement learning and Markov games for multi-agent reinforcement learning.

2.1 MDPs

An MDP is defined by a set of environmental states, $\mathcal{S}$, a set of agent's possible actions, $\mathcal{A}$, a transition probability from a state $s \in \mathcal{S}$ to next state $s' \in \mathcal{S}$, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, a reward function designed for each specific task, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. The discount controls factor controls how much effect future rewards have on the optimal decisions. With small values of γ emphasizing near-term gain and larger values giving significant weight to later reward.

At each time step, the agent is at a state s_t. Following a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, with $\Delta(\cdot)$ denotes probability distributions, the agent plays an action a_t. Given the current state and the action the agent performs, the environment transits into next state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the environment returns an immediate reward $R(s_t, a_t)$ to the agent.

The goal of the agent is to find an optimal policy $\pi^\star : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected sum of discounted reward start from the initial state s_0, defined as $\mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h)|s_0 = s, a_h \sim \pi(\cdot|s_h), s_{h+1} \sim P(\cdot|s_h, a_h)\right]$. Accordingly, one can define the action-value function (Q-function) and state-value function (V-function) under policy π to reason the policy's long-term effect

$$V^\pi(s) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h)|s_0 = s, a_h \sim \pi(\cdot|s_h), s_{h+1} \sim P(\cdot|s_h, a_h)\right] \tag{1}$$

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) | s_0 = s, a_0 = a, a_h \sim \pi(\cdot|s_h), s_{h+1} \sim P(\cdot|s_h, a_h)\right] \tag{2}$$

There always exists a deterministic optimal policy,$\pi^\star$, for infinite horizon discounted MDP, such that $V^{\pi^\star}(s) \geq V^{\pi}(s), \forall s, \forall \pi$. We often denote $V^\star$ in short for $V^{\pi^\star}$[4]. The properties of optimal policy $\pi^\star$ can be described by Bellman Optimality,

$$V^\star(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^\star(s')\right]\right] \tag{3}$$

$$Q^\star(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q^\star(s', a')\right] \tag{4}$$

2.2 MARKOV GAMES

A Markov games can be regarded as a multi-player extension to the MDP definition. However, in a Markov game, the player proceeds by steps from position to position, according agents' joint actions (Fig.1). It is defined by the number of agents, N, a set of states, $\mathcal{S}$, a collection of action sets, $\{\mathcal{A}^i\}_{i \in \{1 \ldots N\}}$. We denote $\mathcal{A} := \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$. A transition probability $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function for each agent, $\{R^i\}_{i \in \{1 \ldots N\}}$, and a discount factor $\gamma \in [0, 1)$.

The goal of each agent is to find an optimal policy $\pi^{i,\star} : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected sum of discounted reward start from the initial state s_0, defined as $\mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r^i(s_h, a_h) | s_0 = s, a_h \sim \pi(\cdot|s_h), s_{h+1} \sim P(\cdot|s_h, a_h)\right]$, where $\pi = \{\pi^i, \pi^{-i}\}$. Again, we can define the value-function $V^{\pi^i} : \mathcal{S} \to \mathbb{R}$ of agent i as

$$V^{\pi^i, \pi^{-i}}(s) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r^i(s_h, a_h) | s_0 = s, a_h \sim \pi(\cdot|s_h), s_{h+1} \sim P(\cdot|s_h, a_h)\right] \tag{5}$$

The Nash Equilibrium in Markov games is defined as a strategy profile $\pi^\star = (\pi^{i,\star}, \pi^{-i,\star})$ if

$$V^{\pi^{i,\star}, \pi^{-i,\star}}(s) \geq V^{\pi^i, \pi^{-i,\star}}(s), \forall s, \forall \pi^i \tag{6}$$

In this project, we consider two agents zero-sum Markov games. This allows us to use a single reward function that agent 1 tries to maximize and agent 2 tries to minimize. Specifically, the value function defined in Eq.5 satisfies $V^{\pi^1, \pi^2} = -V^{\pi^2, \pi^1}$.

Similar as the minimax theorem in zero-sum games, for two- player zero-sum Markov games with finite state and action spaces, one can define the optimal value function $V^\star : \mathcal{S} \to \mathbb{R}$ as[5]

$$V^\star = \max_{\pi_1} \min_{\pi_2} V^{\pi_1, \pi_2} = \min_{\pi_2} \max_{\pi_1} V^{\pi_1, \pi_2} \tag{7}$$

The value of a state $s \in \mathcal{S}$ in a Markov game in an analogue to the Bellman optimality for MDPs is [2]

$$V^\star(s) = \max_{a_1 \in \mathcal{A}^1} \min_{a_2 \in \mathcal{A}^2} \mathbb{E}_{a_1 \sim \pi^{1,\star}(\cdot|s), a_2 \sim \pi^{2,\star}(\cdot|s)} \left[Q^\star(s, a_1, a_2)\right] \tag{8}$$

$$Q^\star(s, a_1, a_2) = r(s, a_1, a_2) + \gamma \mathbb{E}_{s' \in P(\cdot|s, a_1, a_2)} V^\star(s') \tag{9}$$

3 MDPs ALGORITHMS

Value-based RL methods are devised to find a good estimate of the state-action value function, namely, the optimal Q-function $Q^\star$. The (approximate) optimal policy can then be extracted by taking the greedy action of the Q-function estimate.

3.1 VALUE-ITERATION

Define a Bellman operator $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where

$$(\mathcal{T}Q)(s,a) := r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a' \in \mathcal{A}} Q(s', a') \tag{10}$$

Eq.4 shows that $Q^\star$ is a fixed-point solution. The map is also a contraction map [1], i.e., $||\mathcal{T}Q - \mathcal{T}Q'||_\infty \leq \gamma ||Q - Q'||_\infty$. Therefore, we conclude that value-iteration algorithm, $Q^{t+1} \Leftarrow \mathcal{T}Q^t$, converges to the optimal Q-value, $Q^\star$.

The approximate optimal policy can be extracted with $\pi^t(s) = \arg\max_a Q^t(s,a) \to \pi^\star(s) \forall s$.

3.2 Q-LEARNING

In value-iteration algorithm, we assume the agent knows the transition probability. In contrast, Q-learning is a model-free reinforcement learning algorithm. It does not require a model of the environment, and it can handle problems with stochastic transitions and rewards without requiring adaptations.

The agent maintains an estimate of the Q-value function. When transitioning from state-action pair (s_t, a_t) to next state s_{t+1}, the agent receives a payoff r and updates the Q-function according to[6]:

$$Q^{t+1}(s_t, a_t) = (1 - \alpha_t) Q^t(s_t, a_t) + \alpha_t [r(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q^t(s_{t+1}, a')] \tag{11}$$

If $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$, then Q-learning can be proved to converge to the optimal Q-value function almost surely [6] with finite state and action spaces.

4 ZERO-SUM MARKOV GAMES ALGORITHMS

Similar as in single-agent MDPs, value-based methods aim to find an optimal value function, from which the joint Nash equilibrium policy can be extracted.

4.1 MINIMAX VALUE-ITERATION

Define a Bellman operation $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where

$$(\mathcal{T}Q)(s,a) := r(s,a_1,a_2) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a_1,a_2)} \left[\max_{a_1 \in \mathcal{A}^1} \min_{a_2 \in \mathcal{A}^2} \mathbb{E}_{a_1 \sim \pi_1(\cdot|s'), a_2 \sim \pi_2(\cdot|s')} Q(s', a_1, a_2) \right] \tag{12}$$

Eq.4 shows that $Q^\star$ is a fixed-point solution. The map is a contraction map[7], i.e., $||\mathcal{T}Q - \mathcal{T}Q'||_\infty \leq \gamma ||Q - Q'||_\infty$. Therefore, we conclude that minimax value-iteration algorithm, $Q^{t+1} \Leftarrow \mathcal{T}Q^t$, converges to the optimal Q-value $Q^\star$.

The approximate optimal policy can be extracted with $\pi^t(s) = \arg\max_{a_1} \min_{a_2} Q^t(s,a) \rightarrow \pi^\star(s) \forall s$.

4.2 MINIMAX Q-LEARNING

Noticed that value-iteration algorithm is model-based due to the need of computing the Bellman operator defined in Eq. 12. By estimating the Bellman operation via data-driven approximation, [2] has proposed minimax Q-learning, which extends the Q-learning algorithm for single-agent[6] to two-player zero-sum Markov games.

In the t-th iteration, it only updates the value of $Q(s_t, a_t)$ and keeps other entries of Q unchanged. Specifically, we have

$$Q^{t+1}(s_t, a_t^1, a_t^2) = (1-\alpha_t) Q^t(s_t, a_t^1, a_t^2) + \alpha \left[r(s_t, a_t^1, a_t^2) + \gamma \max_{a_1' \in \mathcal{A}^1} \min_{a_2' \in \mathcal{A}^2} Q^t(s_{t+1}, a_1', a_2') \right] \quad (13)$$

Similar to MDPs, if $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$, then Q-learning converges[8] to the optimal Q-value defined by combining Eq. 3 and Eq. 4.

5 Simulation

5.1 Simple stochastic game simulation

In this project, we design a simple game to test the value iteration and Q-learning based method in two-player zero-sum stochastic game. The game contains two states, and each player has two actions to choose at each states. The reward shown in table 1 and table 2 are based on the state and the joint actions of the two players.

Table 1: Reward matrix at state 1

player 1 / player 2	action 1	action 2
action 1	(1,-1)	(0,0)
action 2	(0,0)	(2,-2)

Table 2: Reward matrix at state 2

player 1 / player 2	action 1	action 2
action 1	(-1,1)	(0,0)
action 2	(0,0)	(-2,2)

The transition matrix shown in table is the possibility to transit to state 1 The idea of this game is: when a player is able to get a large instant reward, it will be likely to transit to an unfavorable state. Take player 1 as an example, at state 1, it can get a higher reward if it picks action 2. However, by choosing to play action 2, it will be more likely to go to state 2 where the payoffs are always less than zero. For player 2, it follows the same idea. As a result, for each player, it needs to balance the instant payoff and future payoff. Also, it needs to consider the opponent's action.

Table 3: Reward matrix at state 1

player 1 / player 2	action 1	action 2
action 1	0.4	0.75
action 2	0.6	0.2

Table 4: Reward matrix at state 2

player 1 / player 2	action 1	action 2
action 1	0.6	0.25
action 2	0.4	0.8

The pseudocode for value iteration minimax algorithm is illustrated in Algorithm 1. Firstly, Q and V matrix are initialized, the initial policy was set to be uniform distribution, and the initial value of alpha is 1. At every time step, the players choose their action according to the policy distribution. During the learning process, players update their Q and V values, as

well as their policy by optimizing the minimax function. In addition, alpha will decrease at each step.

Algorithm 1: The Value Iteration Minimax Algorithm

initialization:
Q[s,a_1,a_2] := 1, V[s] := 1, pi[s,a_1] := 1, $\alpha := 1$ **for** all s in S, a_1 in A_1, a_2 in A_2;
Choose an action:
Choose an action for every player with probability pi[s,a_1].
Learn:
while *True* **do**
  Choose reward **rew** corresponding to state and actions of of every agent;
  Q[s,a_1,a_2] = (1-α)*Q[s,a_1,a_2]+α*(rew+$\gamma \sum_{s'}$P[s,a_1,a_2,s']*V[s']);
  pi[s,.] = $\arg\max_{pi'[s,.]} \min_{a'_2} \sum_{a'} pi'[s,.] * Q[s, a'_1, a'_2]$;
  V[s] = $\min_{a'_2} \sum_{a'_1} pi'[s,.] * Q[s, a'_1, a'_2]$;
  α = α*decay;
end

The value V, the prediction for the future reward starting from each state, is shown in Figure 2. The red line is the expected value for the Nash equilibrium. According to the definition of V, the V values predicted by player 1 and player 2 in a zero-sum game should be exactly opposite. The plots show that the V values, predicted by player 1 and player 2, converges after around 400 iteration. The result is intuitive, as for state 1, the payoffs for player 1 are always great or equal to 0, and the V value for player 1 starting at state 1 is greater than 0. Since the payoffs at the two states have the same absolute value, the V of state 1 and state 2 also have the same absolute value as expected.

As shown in Figure 3 The policy for each player converges to an equilibrium. The final policy shows that both players do not always choose the action that can bring a higher instant payoff given that they are more likely to transit to a state with a much lower reward, as we expected.

Furthermore, we analyse how $decay$ and γ influence the learning performance. When $decay$ is increased, the update at every learning step is more significant and the game converges faster. As shown in Figure 4, with a small increment of the $decay$ value, the V value estimated by player 1 still converge to the same value but in a slower speed.

For the discount factor, γ, it controls how much the agents take the future reward into account. When γ is increased, as shown in Figure 5, the agents consider more about the future value and it converges in a much slower speed.

We also simulate the same game by minimax-Q learning. As shown in algorithm 2, the difference between Q-learning and value iteration is that Q-learning is model free, which means it doesn't know the transition possibility matrix during the learning process. Instead, it updates the Q value only based on the next state and current reward returned from the environment.

The value V obtained from the minimax-Q learning is shown in Figure 6. It converges to the same equilibrium as the value iteration method. However, it takes more iterations to converge and there are more oscillations during the learning process. It takes about 1000 iterations to reach the equilibrium value. The Figure 7 demonstrates that the policy converge to the same optimal policy as value iteration method.

Algorithm 2: The Minimax-Q Algorithm.

initialization:
Q[s,a_1,a_2] := 1, V[s] := 1 , pi[s,a_1] := 1/$|A_1|$, $\alpha := 1$ **for** all s in S, a_1 in A_1, a_2 in A_2;
Choose an action:
Choose an action for every player with probability pi[s,a_1].
Learn:
while *True* **do**
  Choose reward **rew** corresponding to state and actions of every agent;
  Q[s,a_1,a_2] := (1-α)*Q[s,a_1,a_2]+α*(rew+γV[s']);
  pi[s,.] = $\arg\max_{pi'[s,.]} \min_{a'_2} \sum_{a'_1} pi'[s,.] * Q[s,a'_1,a'_2]$;
  V[s] := $\min_{a'_2} \sum_{a'_1} pi'[s,.] * Q[s,a'_1,a'_2]$;
  α := α*decay;
end

5.2 Ball game simulation

In this part, we designed a more sophisticated game, a soccer game, to see how these algorithms work. The game includes a player ready to shoot, player, and a goal keeper, player 2. There are two states: state 1 represents the defender is right in front of the play 1, and state 2 represents the defender mismatched with the offender. At each state, each player can choose to stay or move. The objective of the offender is to find an opportunity to shoot. The objective of the defender is to block the offender.

Table 5: Reward matrix at state 1

player 1 / player 2	stay	move
stay	(-0.75,0.75)	(-0.8,0.8)
move	(-0.9,0.9)	(-2,2)

Table 6: Reward matrix at state 2

player 1 / player 2	stay	move
stay	(4,-4)	(0.85,-0.85)
move	(0.9,-0.9)	(0.6,-0.6)

Table 7: Reward matrix at state 1

player 1 / player 2	stay	move
stay	(-0.75,0.75)	(-0.8,0.8)
move	(-0.9,0.9)	(-2,2)

Table 8: Reward matrix at state 2

player 1 / player 2	stay	move
stay	(4,-4)	(0.85,-0.85)
move	(0.9,-0.9)	(0.6,-0.6)

The payoff matrix for the soccer game is shown in table 5 and table 6. In state 1, if the two players are right in front of each other, the defender would like to remain in the position, but the offender would want to switch a position. Therefore, player 2 has higher payoff. In state 2, player 1 prefers to stay in this state, since it has a clear shot. However, for player 2, it would like to switch a position in order to block the ball. Therefore, player 1 in this state has higher payoff. The transition matrix defined in table 7 and table 8 indicates that when they both choose to move or to stay, they will be more likely to stay at their current state.

We trained the optimal policy with the value iteration minimax and minimax Q-learning algorithms, and the results are shown in Figure 8 and Figure 9. The average reward for player 1 after 100,000 repeated experiments is shown in table 9. Comparing the reward of player 1 when both players choose the optimal policies and the that of player 1 when they choose to move at every step, we observe that both players benefit from the optimal policy.

Table 9: Reward matrix at state 1

player 1 / player 2	optimal policy	naive policy
optimal policy	0.859	1.189
naive policy	0.638	-

6 DISCUSSION

This project analyzes the MDPs and the Markov games formalism as mathematical frameworks for reasoning about single-agent and multi-agent environments. In particularly, the project describes value-based reinforcement approaches to find the optimal Q-values in MDPs and two-player zero-sum Markov game, and then extract the optimal policy from it.

For MDPs, we first introduce value-iteration algorithm, in which we assume the RL agent know how the environment works by having the knowledge of the transition function. Then, we talk about a model-free algorithm, Q-learning, in which the agent does not know how the environment works and it can learn how to behave optimally through purely interacting with the environment. Both algorithms build upon fixed-point solution demonstrated by Bellman optimality. The mappings are also shown to be contraction maps. From both, we conclude that the value-iteration algorithm and the Q-learning algorithm converge to the optimal Q-value.

Building upon the definitions and algorithms discussed for MDPs, we explore value-based algorithms for two-player zero-sum Markov games, namely the minimax value-iteration and the minimax Q-learning. Similar to the MDPs settings, minimax value-iteration is a model-based algorithm, while minimax Q-learning is a model-free algorithm. The Bellman equation again shows that the optimal value function is an unique fixed point solution, which can be obtained via dynamic programming type methods.

Another type of RL algorithms directly searches over the policy space, which is called policy-based algorithms. Policy-based methods approximate the optimal policy by employing parameterized function approximators such as neural networks or softmax policy, denoted as $\pi(\cdot|s) \approx \pi_\theta(\cdot|s)$. The optimal policy, characterized by the optimal parameters, can be found by applying stochastic gradient descent approach directly on the policy. We would like to further explore the mathematical foundation for policy-based methods in MDPs and extend them to Markov games settings. Furthermore, we would also want to study on other game settings in multi-agent environments not only restricted to two-player zero-sum Markov games.

Acknowledgement

We would like to thank our instructor Professor Francesca Parise, who guided us on this final project, by suggesting workable project topics, providing related papers, and guiding us through the complex proof demonstrated in papers.

References

[1] L. S. Shapley. "Stochastic Games". In: *Proceedings of the National Academy of Sciences* 39.10 (1953). Publisher: National Academy of Sciences _eprint: https://www.pnas.org/content/39/10/1095.full.pdf, pp. 1095–1100. ISSN: 0027-8424. DOI: 10.1073/pnas.39.10.1095. URL: https://www.pnas.org/content/39/10/1095.

[2] Michael L. Littman. "Markov Games as a Framework for Multi-Agent Reinforcement Learning." In: *ICML*. Ed. by William W. Cohen and Haym Hirsh. Morgan Kaufmann, 1994, pp. 157–163. ISBN: 1-55860-335-2. URL: http://dblp.uni-trier.de/db/conf/icml/icml1994.html#Littman94.

[3] Yaodong Yang and Jun Wang. "An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective". In: *CoRR* abs/2011.00583 (2020). arXiv: 2011.00583. URL: https://arxiv.org/abs/2011.00583.

[4] Alekh Agarwal et al. *Reinforcement Learning: Theory and Algorithm*. 2021.

[5] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms". In: *CoRR* abs/1911.10635 (2019). arXiv: 1911.10635. URL: http://arxiv.org/abs/1911.10635.

[6] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: *Machine Learning* 8.3 (May 1992), pp. 279–292. ISSN: 1573-0565. DOI: 10.1007/BF00992698. URL: https://doi.org/10.1007/BF00992698.

[7] Michail G. Lagoudakis and Ronald Parr. "Value Function Approximation in Zero-Sum Markov Games". In: *CoRR* abs/1301.0580 (2013). arXiv: 1301.0580. URL: http://arxiv.org/abs/1301.0580.

[8] Csaba Szepesvári and Michael L. Littman. "A Unified Analysis of Value-Function-Based Reinforcement-Learning Algorithms". In: *Neural Computation* 11.8 (Nov. 1999). _eprint: https://direct.mit.edu/neco/article-pdf/11/8/2017/814264/089976699300016070.pdf, pp. 2017–2060. ISSN: 0899-7667. DOI: 10.1162/089976699300016070. URL: https://doi.org/10.1162/089976699300016070.

Appendix

Appendix 1: Bellman equations

One can write the V-function and Q-function in a recursive way, in the benefit of dynamic programming. Bellman equation for V-function is

$$V^{\pi}(s) = \max_{a' \in \mathcal{A}} Q^{\pi}(s, a') \tag{14}$$

and Bellman equation for Q-function is

$$Q^{\pi}(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^{\pi}(s) \right] \tag{15}$$

Appendix 2: Simulation figures

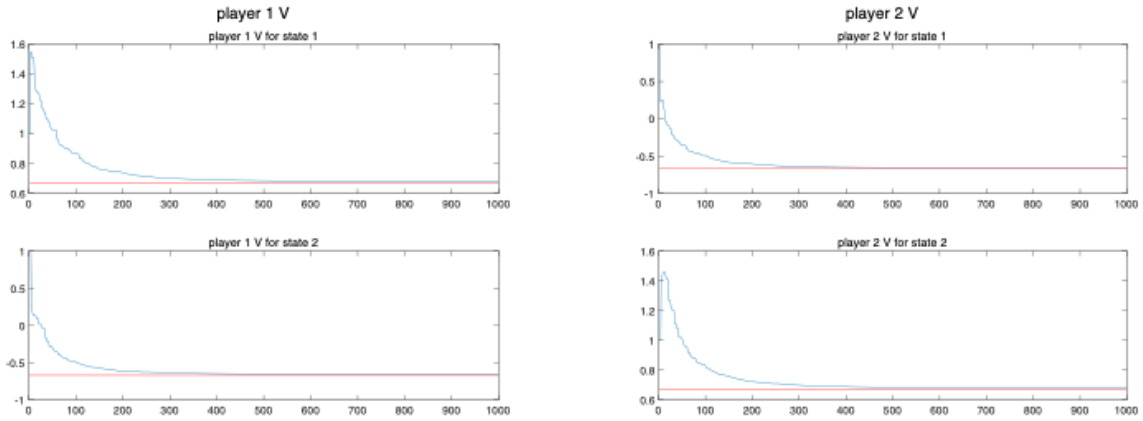


Figure 2: V value during the value iteration training process

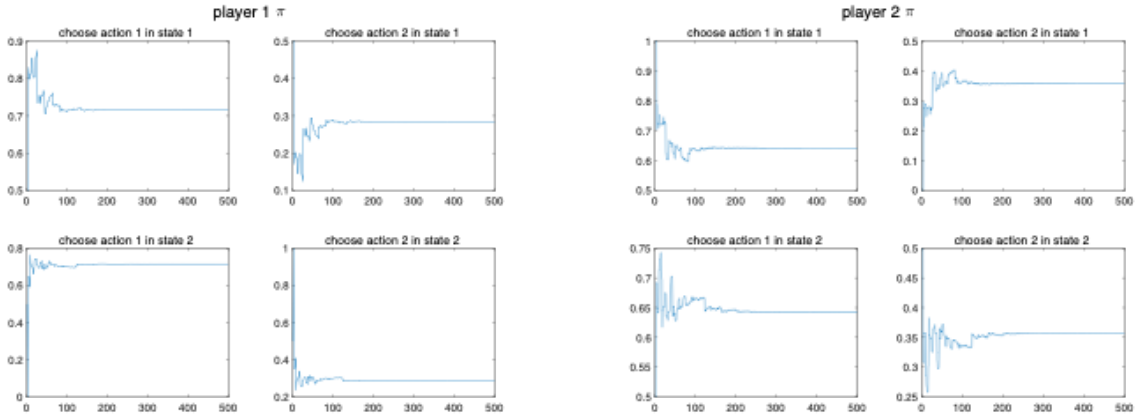
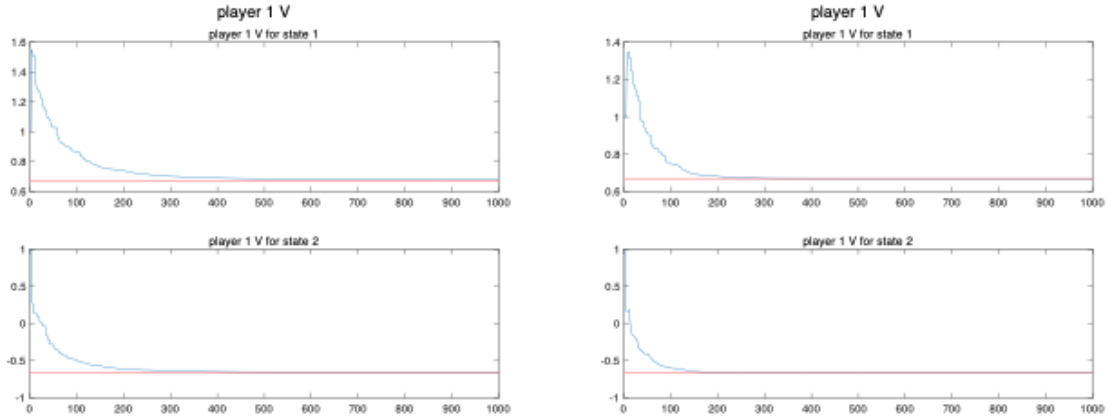
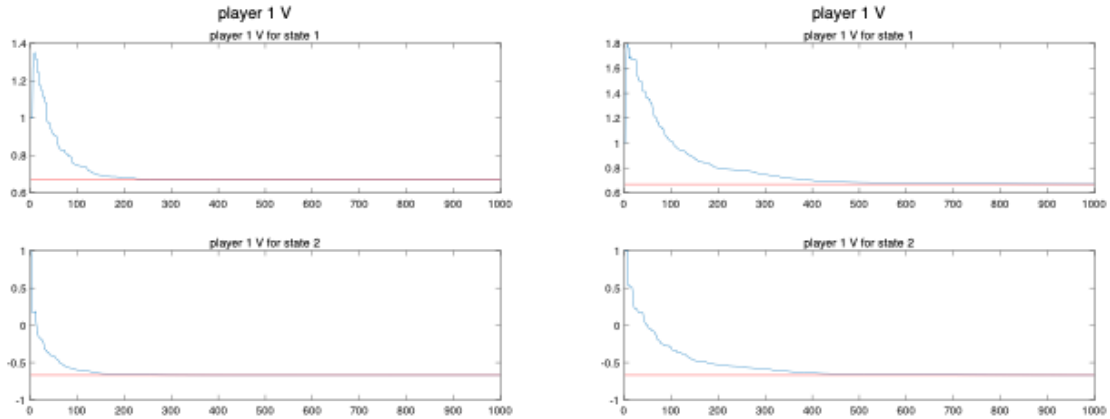


Figure 3: Policy during the value iteration training process



(a) Decay = 0.995 (b) Decay = 0.999

Figure 4: Player 1 V value comparison for different Decay value



(a) $gamma = 0.8$ (b) $gamma = 0.9$

Figure 5: Player 1 V value comparison for different γ

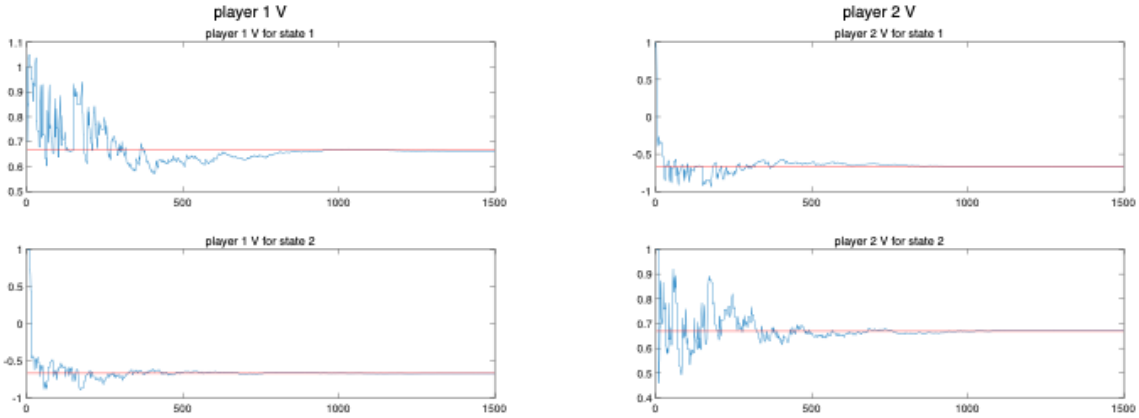


Figure 6: V value during the minimax-Q training process

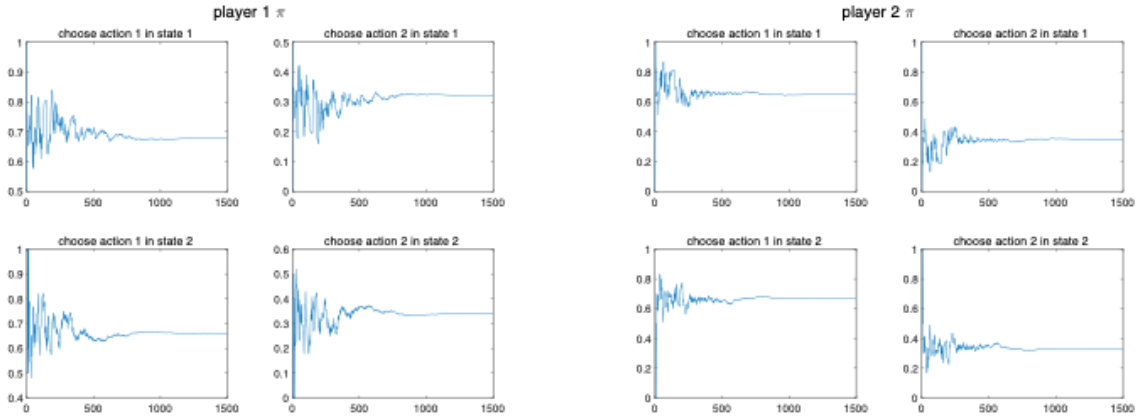


Figure 7: policy during the minimax-Q training process

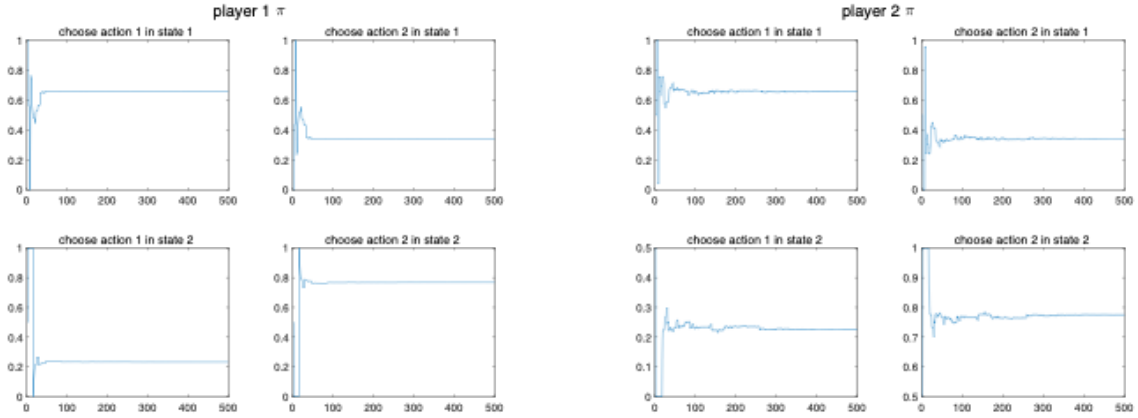


Figure 8: policy for the ball game trained by value iteration algorithm

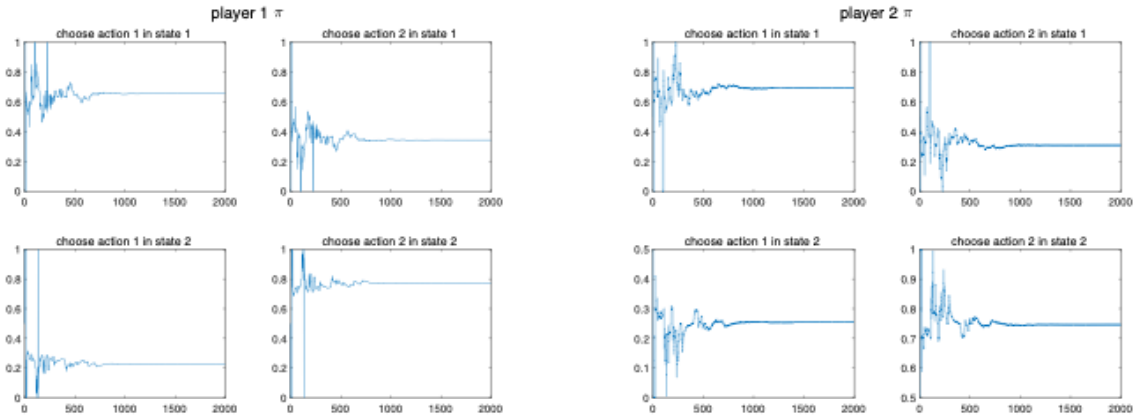


Figure 9: policy for the ball game trained by Minimax-Q algorithm