Comparing Multi Agent Reinforcement Learning (MARL) algorithms

Álvaro Menéndez , Joaquín Monedero

February 2024

Introduction

Muli Agent Reinforcement Learning (MARL) studies how multiple agents, each with its own objective, interact in a common environment. MARL has proven to solve complex real-world problems, such as network resource allocation and sharing, network routing, and traffic signal controls [1]. In this document we compare two MARL algorithms, namely **Policy Hill Climbing** and **Win or Learn Fast (WoLF)** in some simple 2×2 Normal Form games.

Implementation

The agents were implemented using Java. The games in which the agents were tested had the following payoff matrices:

Table 1: Payoff Matrices

		Player 2		Game
		Action A	Action B	
Player 1	Action A	2, 1	0, 0	Battle of the Sexes
	Action B	0, 0	1, 2	
	Action A	1, -1	-1, 1	Matching Pennies
	Action B	-1, 1	1, -1	
	Action A	-3, 3	0, 5	Prisoner's Dilemma
	Action B	-5, 0	-1, -1	

Policy Hill Climbing

The Policy Hill Climbing (PHC) continually moves to the action with a highest Q value. The agent is initialized by setting $Q(s,a) = 0$, $\pi(s,a) = \frac{1}{|A|}, \ \forall s, a$

Where:

1. s is a state, a is an action
2. $Q(s,a)$ is the Q-value function
3. $\pi(s,a)$ denotes the probability of selecting action a in state s
4. A is the set of actions.

In the main loop, an action a is chosen based on the probability density function (PDF) π and the reward given that action a is used to update $Q(s,a)$ as follows:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s', a')\right]$$

Note that for simplicity, we decided to let γ (the discount factor) equal to 0 which makes the agent short-sighted by only considering current rewards. Then, π is updated by adding δ to the action with highest current Q value and normalising it.

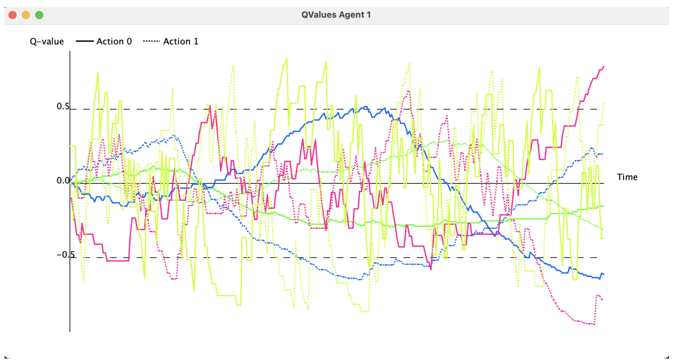
Win Or Lose Fast (WoLF)

This policy is similar to the PHC but with two learning rates: a small δ_w for when it is winning, and a larger δ_l for when it needs to learn fast. The choice between these learning rates depends on the policy's current performance. If it is doing well compared to an average performance, δ_w is chosen. If the policy is underperforming, δ_l is chosen to catch up faster.

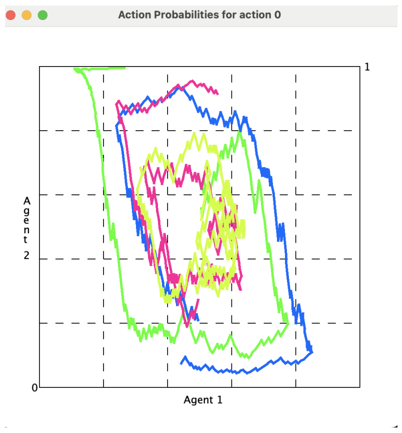
Experiments & Results

After implementing the agents in the source code environment, an experiment was done in order to discover the effect of changing the α parameter. This parameter is the learning rate and the convention is that the values range from 0 to 1. If the learning rate is 0, then the agent will not learn from new information. On the other hand, if the value is 1, then the agent will completely override old information with new information. This being said, it is important to find a value in between 0 and 1, that will help the agent learn enough new information without ignoring the already known information. Ideally, we want to find a parameter that will make the agent behave according to a Nash Equilibrium.

In order to find an appropriate value, an experiment was performed as follows: The PHC agent played the **Battle of Sexes** game **five** times against an independent Q-learning agent, using the the following α values: `{0.05, 0.1, 0.25, 0.5}`. After using each value for α, the following plots were created, where we can see the Q-values over time for the PHC agent as well as the action 0 probabilites over time. Note that in this game, there are two nash equilibria at (0, 0) and (1, 1), and the Q-learning agent was initialised with probabilities (0.4, 0.6) so that they converged into (1, 1).



(a) Q-values over time

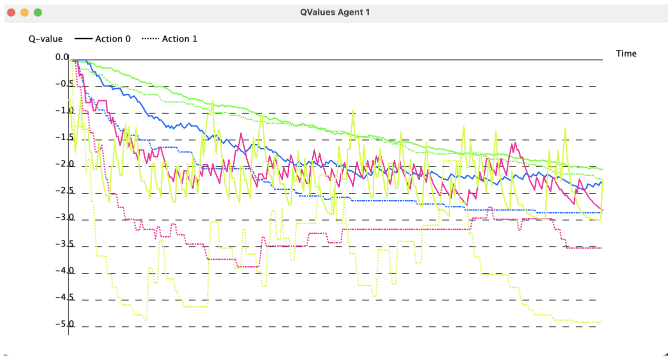


(b) Action probabilities over time

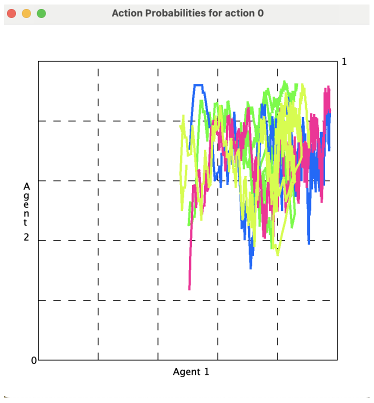
Figure 1: Policy hill climbing agent vs Q-learner Agent with $\alpha = 0.5$ (Matching Pennies)

After taking a closer look at said plots, we can see that the ideal value for α is 0.5 as it is the only value were the the graph eventually convergences into a Nash equilibrium. When using the lower values of α we can see the Q-values oscillate between negative and positive values, without converging to a Nash Equilibrium.

Secondly, we used the same PHC agent with these parameters for the Prisoner's Dilemma and the Battle of sexes games. The goal was to investigate if the agent also managed to learn to behave according to a Nash Equilibrium. The same graph was plotted for both games. From these graphs we can see that the agent is behaving consistently across the 3 different games. When playing the battle of sexes games, the Q-values rapidly converge into a Nash Equilibrium with a positive value. Oppositely, when playing the Prisoner's Dilemma game the graph converges to a negative value.

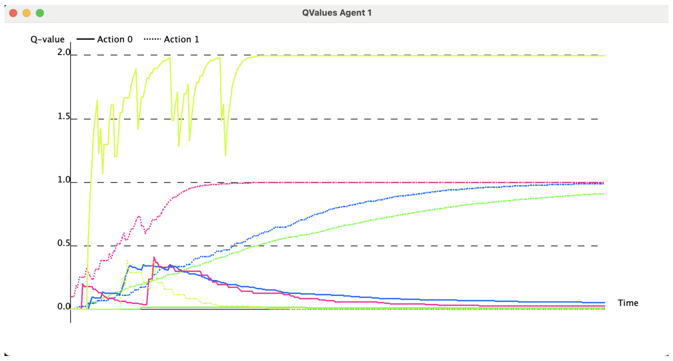


(a) Q-values over time

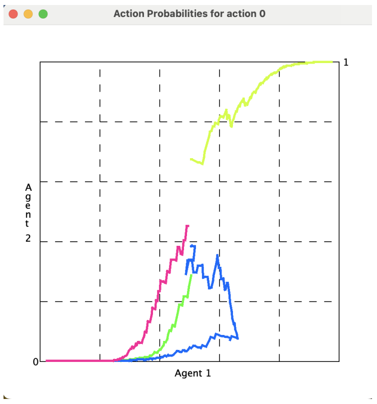


(b) Action probabilities over time

Figure 2: Policy hill climbing agent vs Q-learner Agent with $\alpha = 0.5$ (Prisoners Dilemma)



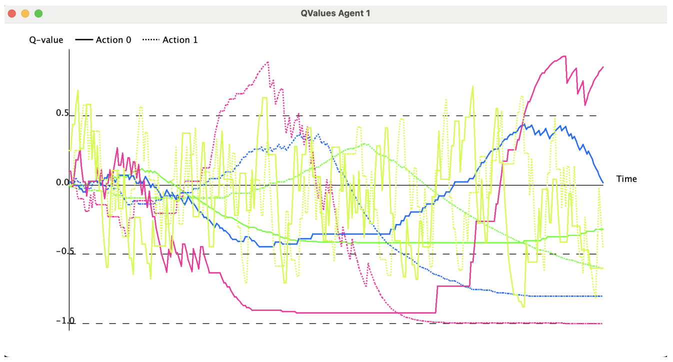
(a) Q-values over time



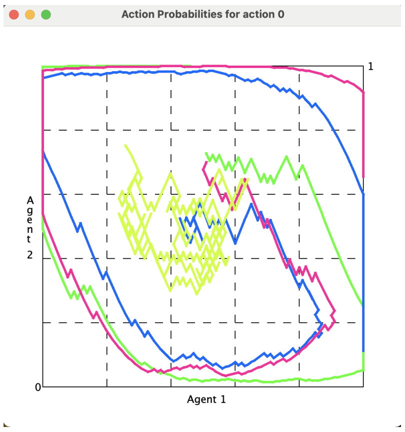
(b) Action probabilities over time

Figure 3: Policy hill climbing agent vs Q-learner Agent with $\alpha = 0.5$ (Battle of Sexes)

Given that the PHC and WoLF agents are very similar, and for simplicity, we checked if the WoLF behaved similarly with an α value of 0.5 when playing the 3 games against the provided Q-learning agent. Below we have included the graph of Q-values for the WoLF agent over time, for the 3 different games. From these graphs we can conclude that the α value of 0.5 is also an appropriate value, as we see the exact same behaviour as we saw when experimenting with the PHC agent.

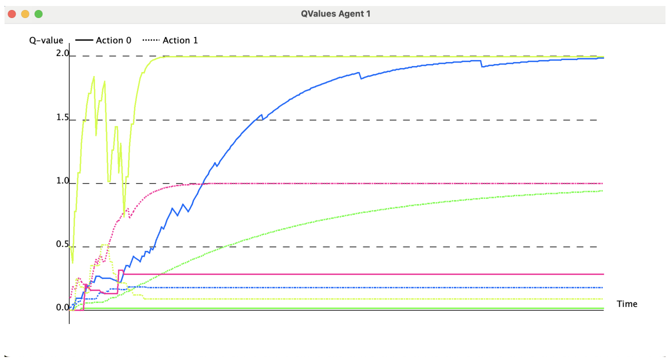


(a) Q-values over time

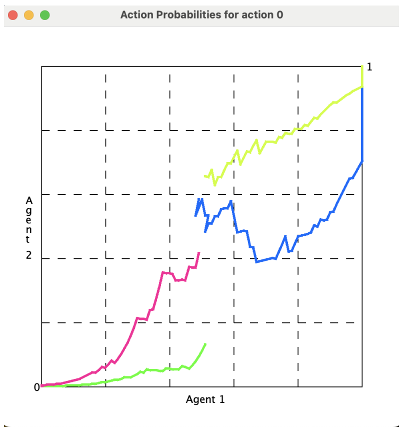


(b) Action probabilities over time

Figure 4: Win Or Lose Fast vs Q-learner Agent with $\alpha = 0.5$ (Matching Pennies)

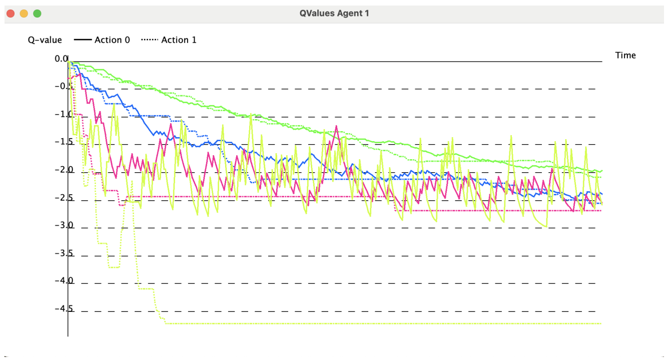


(a) Q-values over time

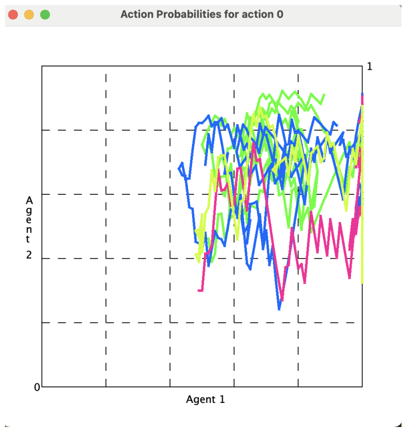


(b) Action probabilities over time

Figure 5: Win Or Lose Fast vs Q-learner Agent with $\alpha = 0.5$ (Battle of Sexes)



(a) Q-values over time



(b) Action probabilities over time

Figure 6: Win Or Lose Fast vs Q-learner Agent with $\alpha = 0.5$ (Prisoners Dilemma)

References

[1] Abdikarim Mohamed Ibrahim et al. “Applications of Multi-Agent Deep Reinforcement Learning: Models and Algorithms”. In: *Applied Sciences* 11.22 (2021). ISSN: 2076-3417. DOI: 10.3390/app112210870. URL: https://www.mdpi.com/2076-3417/11/22/10870.