Adaptive Agents in the Iterated Prisoner's Dilemma: A MARL Approach with DQNs

Luca Burroni
Autonomous and Adaptive Systems Course
University of Bologna, Department of Computer Science and Engineering
luca.burroni@studio.unibo.it

Abstract

In this project, we explore the application of Multi-Agent Reinforcement Learning (MARL) with Deep Q-Networks (DQNs) to study adaptive agents in the context of the Iterated Prisoner's Dilemma (IPD). The primary goal of this project is to investigate how different learned agents' behaviors change in response to variations in environmental parameters. We provide an overview of our implementation, present experimental results, and discuss key findings.

1 Introduction

Understanding human cooperation and competition has long been a central pursuit in the fields of game theory and social science. At the heart of this exploration lies the concept of matrix game social dilemmas, which provide a way to examine complex decision-making processes. These dilemmas are characterized by specific payoff matrices, as shown in tables below.

	C	D
C	R, R	S, T
D	T, S	P, P

Prisoner	C	D
C	3, 3	0, 5
D	5, 0	1, 1

The two actions are cooperate and defect. The four possible outcomes are R (reward of mutual cooperation), P (punishment arising from mutual defection), S (sucker outcome obtained by the player who cooperates with a defecting partner), and T (temptation outcome achieved by defecting against a cooperator). As per *Learning Dynamics in Social Dilemmas* (Macy and Flache, 2002) using these payoffs, we can define a social dilemma as any ordering of these four payoffs such that the following four conditions are satisfied:

R > P: In this condition, players express a preference for mutual cooperation (CC) over mutual defection (DD).

R > S: This condition underscores the preference for mutual cooperation (CC) over unilateral cooperation (CD).

2R > T + S: Here, the emphasis shifts to the players' preference for mutual cooperation over an equal probability of unilateral cooperation and defection.

T > R: In this case, players lean towards unilateral defection (DC) over mutual cooperation, often driven by greed, or they might favor P > S, indicating a preference for mutual defection over unilateral cooperation out of fear.

In the world of matrix game social dilemmas, the Iterated Prisoner's Dilemma (IPD) stands as an exemplar. It is a classic game theory scenario where two players repeatedly confront the choice between cooperation and betrayal. Each round of interaction presents the same dilemma, offering opportunities for strategies to evolve over time.

Preprint. Under review.

In this project we will use Multi-Agent Reinforcement Learning (MARL) methods in an IPD environment. Through a series of experiments, we try to study different behaviors that adaptive agents learn under different environmental parameter values.

2 Methodology

2.1 Environment

Since OpenAI Gym does not provide an IPD environment we developed a custom one (Gym environment). This environment allows two agents to interact in a repeated IPD scenario. Possible actions are cooperate (0) and defect (1).
The observation space is designed to encapsulate the historical actions of both players, offering a valuable context for decision-making. It is represented by an array with size determined by the "max_history_length" parameter. This encoded history equips agents with a chronological understanding of the game, enabling them to recognize patterns and make informed choices.
In order to pursue the goal of the project the environment parameters max_history_length, max_rounds and payoff_matrix can be passed as inputs to the environment. max_rounds parameter represents the max number of rounds per episode.

2.2 Agents

We developed the DQNAgent class, implementing a Deep Q-Network (DQN) as the central decision-maker within the Iterated Prisoner's Dilemma (IPD). The DQN algorithm, as introduced in *Human-level control through deep reinforcement learning* (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, Petersen, Beattie, Sadik, Antonoglou, King, Kumaran, Wierstra, Legg, and Hassabis, 2015), is used to train our agents.
Each player is instantiated as a single DQN agent. To select actions, an epsilon-greedy policy with a decay strategy is employed. During agent updates, a replay buffer is used to store and sample experiences.
The DQN agents are trained over a specified number of episodes. During training, they explore the environment using the epsilon-greedy policy, gradually shifting towards exploiting their learned Q-values. The agents aim to maximize their own expected rewards. Evaluation metrics such as average episode rewards or convergence plots are used to assess their performance.
The core update rule used for Q-value updates in our DQN agent is as follows:
$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha\left[r_i + \gamma \max_{a' \in A_i} Q_i(s',a') - Q_i(s,a)\right]$
This formula represents how the Q-values are updated to improve the agent's decision-making over time.
Additionally, the neural network (NN) model used in our DQNAgent consists of an input layer with an input shape corresponding to the problem's state space, followed by two hidden layers. Each hidden layer consists of 32 units with ReLU activation functions, allowing the NN to capture intricate state-action relationships. The output layer, with a size equal to the number of possible actions (n_outputs), produces Q-value estimates, serving as the function approximator for the Q-values.

3 Experiments and Results

In order to conduct many different experiments, changing each time one or more parameter, the code is quite modular. To run an experiment is sufficient to call the function run_experiment(...) that accept many parameters in input and call all others functions.
The function input parameters are:

- Environment parameters:
  - max_history_length (default 16)
  - max_rounds (default 24)
  - payoff_matrix (default [[[3, 3], [0, 5]], [[5, 0], [1, 1]]])
- Training and Evaluation parameters:
  - num_episodes (default 25)
  - batch_size (default 16)

- update_target_network_freq (default 5)
- starting_epsilon (default 0.4)
- num_evaluation_episodes (default 50)

At the end of training phase is plotted a graph with average reward for episode for both agents, is also plotted the average cumulative reward for completeness (even if not really significant).
At the end of evaluation phase are printed some useful insights that show agents learned behavior:

- Average Reward for both players and cumulative (just for completeness)
- Cooperative and Defective percentage of each player
- Combinations [CC, CD, DC, DD]: how often (percentage) are these combinations of actions occurred

3.1 Standard parameters

3.1.1 Expectations

With standard parameters we expect agents to initially explore different strategies, which might include both cooperation and defection. Then average reward for both agents may stabilize and converge to a relatively consistent value, reflecting a balance of cooperation and defection.

3.1.2 Results

As we can see in Figure 1 and Figure 2 the results pretty much match the expectations.

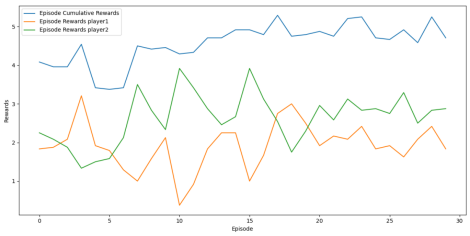

Figure 1: Average rewards per episode with standard parameters

Average Cumulative Reward	4.25
Average player1 Reward	1.92
Average player2 Reward	2.33
Cooperative Percentage player1	45.83%
Cooperative Percentage player2	37.5%
Defective Percentage player1	54.17%
Defective Percentage player2	62.5%
Combinations	
CC	12.5%
CD	25%
DC	33.33%
DD	29.17%

Figure 2: Evaluation metrics whith standard parameters

3.2 Higher payoff for both cooperating

Here we set a higher payoff for both agents cooperating case, in particular we pass as payoff_matrix [[[4, 4], [0, 5]], [[5, 0], [1, 1]]].

3.2.1 Expectations

In this case we expect both agents after some training to converge to always cooperate policy or similar.

3.2.2 Results

As we can see in Figure 3 and Figure 4 the results perfectly match the expectations, in fact both agents in evaluation (after training) use always cooperate policy.

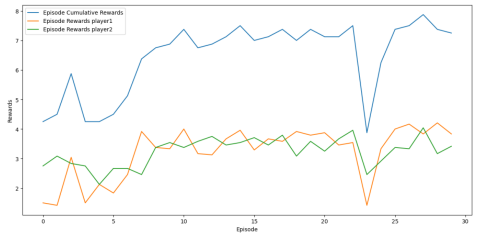

Figure 3: Average rewards per episode with higher (C,C) payoff

Average Cumulative Reward	8.00
Average player1 Reward	4.00
Average player2 Reward	4.00
Cooperative Percentage player1	100%
Cooperative Percentage player2	100%
Defective Percentage player1	0%
Defective Percentage player2	0%
Combinations	
CC	100%
CD	0%
DC	0%
DD	0%

Figure 4: Evaluation metrics with higher (C,C) payoff

3.3 Lower payoff for both cooperating and higher for temptation outcome

We still modify payoff_matrix, but this time, differently from before, we use a higher temptation outcome and a lower (C,C) payoff. the payoff_matrix this time is [[[2, 2], [0, 6]], [[6, 0], [1, 1]]].

3.3.1 Expectations

Inversely from the experiment before this time we expect agents to defect more, but not to converge to always defect policy, because (D,D) payoff is still low.

3.3.2 Results

This time, as we can see in Figure 5 and Figure 6, the results are hard to be interpreted. More or less matches what we expected in a sense that agents do not converge to (C,C) but try to defect the other. It ends in this case in one agent rewards being a lot higher.

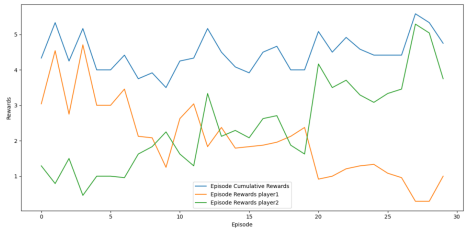

Figure 5: Average rewards per episode with higher T outcome and lower (C,C) payoff

Average Cumulative Reward	5.83
Average player1 Reward	0.04
Average player2 Reward	5.79
Cooperative Percentage player1	95.83%
Cooperative Percentage player2	0%
Defective Percentage player1	4.17%
Defective Percentage player2	100%
Combinations	
CC	0%
CD	0%
DC	95.83%
DD	4.17%

Figure 6: Evaluation metrics with higher T outcome and lower (C,C) payoff

3.4 Higher max_rounds

In this experiment we use a higher max_rounds parameter, means that there will be more step in each episode.

3.4.1 Expectations

A higher max_rounds parameter may lead agents to consider more long term consequences.

3.4.2 Results

Also in this case results are really hard to be interpreted, looking at Figure 7 we can see how, with respect to standard parameters case in Figure 1, in training seems more stable in converging to consistent reward average values. Agents as in standard case reflect a balance in cooperation and defection.

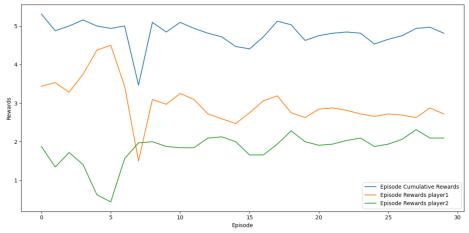

Figure 7: Average rewards per episode with higher max_rounds

Average Cumulative Reward	4.88
Average player1 Reward	2.91
Average player2 Reward	1.97
Cooperative Percentage player1	46.88%
Cooperative Percentage player2	65.62%
Defective Percentage player1	53.12%
Defective Percentage player2	34.38%
Combinations	
CC	33.33%
CD	54.17%
DC	29.17%
DD	16.67%

Figure 8: Evaluation metrics with higher max_rounds

3.4.3 Higher max_history_length

We obtained almost the same results with higher max_history_length, in that case agents have more knowledge about the past, so it still may lead to consider more long term consequences.

3.5 Lower starting epsilon value (less exploration)

In this experiment we use a lower (0.1) epsilon value in order to let agents explore less.

3.5.1 Expectations

Less exploration should lead agents training to converge faster to stable policies.

3.5.2 Results

Results perfectly match the expectations, agents policies converge quickly to always cooperate policy, as we can see in Figure 9 and Figure 10.

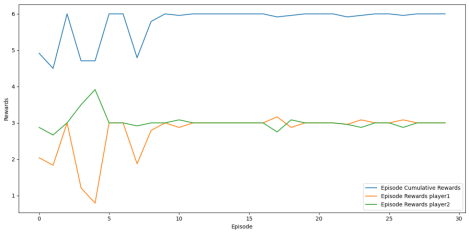

Figure 9: Average rewards per episode with lower starting epsilon value

Average Cumulative Reward	6.00
Average player1 Reward	3.00
Average player2 Reward	3.00
Cooperative Percentage player1	100%
Cooperative Percentage player2	100%
Defective Percentage player1	0%
Defective Percentage player2	0%
Combinations	
CC	100%
CD	0%
DC	0%
DD	0%

Figure 10: Evaluation metrics with lower starting epsilon value

3.6 Higher starting epsilon value (more exploration)

Inversely from before, in this experiment we use a higher (0.8) starting epsilon value in order to let agents explore more.

3.6.1 Expectations

More exploration should lead agents training to converge very slow to stable policies. So we expect a balance of cooperation and defection.

3.6.2 Results

Results pretty much match the expectations, agents policies do not converge to a stable policy, but continue exploring different complex policies leading to a balance of cooperation and defection as we can see in Figure 11 and Figure 12.

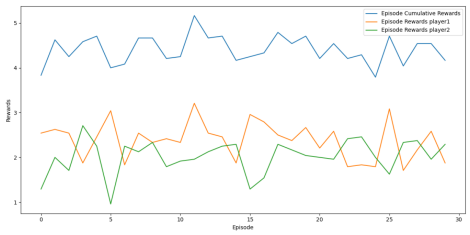

Figure 11: Average rewards per episode with higher starting epsilon value

Average Cumulative Reward	5.29
Average player1 Reward	4.00
Average player2 Reward	1.29
Cooperative Percentage player1	41.67%
Cooperative Percentage player2	95.83%
Defective Percentage player1	58.33%
Defective Percentage player2	4.17%
Combinations	
CC	41.67%
CD	54.17%
DC	0%
DD	4.17%

Figure 12: Evaluation metrics with higher starting epsilon value

4 Link to external resources:

- Gymnasium Documentation
- The project's Git-Hub repository.

References

Dr Macy and Andreas Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3:7229–36, 06 2002. doi: 10.1073/pnas.092080099.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015. doi: 10.1038/nature14236.