Conquering the Evolutionary Multi-Prisoner's Dilemma with Reinforcement Learning

Sangmin Lee[*]
School of Computing
KAIST
alex6095@kaist.ac.kr

Dongwoo Won [*]
School of Computing
KAIST
aiwwdw@kaist.ac.kr

Seungwan Kang [*]
School of Computing
KAIST
seungwanwin@kaist.ac.kr

Abstract

The Evolution of Trust is a multiplayer game based on the prisoner's dilemma. This environment serves as a social experiment to illustrate choices between individual gains and collective benefits. In this study, we investigate the potential for reinforcement learning models to be trained towards pursuing collective benefits within such an environment. Using Q-learning and DQN-based models, we examine the emergence of cooperative behavior and identify models capable of finding optimal solutions. Our findings demonstrate the feasibility of training reinforcement learning agents to maximize rewards while fostering cooperative strategies.

1 Introduction

The Prisoner's Dilemma is a quintessential example in game theory that explores the decision-making process of two individuals who must choose between cooperation and betrayal. If one betrays while the other cooperates, the betrayer gains the maximum benefit while the cooperator receives the least. Conversely, if both betray, they receive moderate benefits. This dilemma effectively illustrates the conflict between individual and collective interests.

"The Evolution of Trust" extends the 1 vs 1 Prisoner's Dilemma game into a group-based social experiment. While Iterated Prisoner's Dilemmas involve multiple rounds of the game between two players, "The Evolution of Trust" conducts Iterated Prisoner's Dilemmas among multiple players. For each round, the ranking of players is determined by the total rewards accumulated. Lower-ranked players are replaced by higher-ranked players' strategies, and the game continues until only one type of strategy survives. At this point, copycat strategy is one of the most successful strategies. It operates by mimicking the opponent's previous action. This simple rule promotes trust and cooperation in repeated game scenarios.

In this study, we aim to identify the optimal strategy for winning the game "The Evolution of Trust" through reinforcement learning. We analyzed whether an agent can learn to cooperate and quickly adapt to the characteristics of opponent players. Initially, we observed the behaviors of agents in a one to one setting before conducting one to many experiments. Starting with basic Q-learning, we implemented and experimented with variations of Q-learning and Deep Q-Networks (DQN). Through these experiments, we observed that models can learn cooperation is a superior action. Additionally, we were able to assess the learning speed and scope of learning for each model.

2 Preliminaries

The Evolution of Cooperation was originally conducted as a study in political science and game theory rather than reinforcement learning. Robert Axelrod conducted experiments within the environment of The Evolution of Trust using models created by individuals from various fields, advocating firm

[*]Equal Contribution

Preprint. Under review.

cooperation as one of the conditions for victory [2]. The Copycat model was demonstrated to be powerful.

Reinforcement learning has been widely applied to situations requiring human decision-making. The Q-learning algorithm decides actions based on the expected value of rewards obtained by taking specific actions in given states [5]. Deep Q-Networks (DQN), which are models that train neural networks based on Q-learning, have expanded the range of states and enabled more stable learning compared to simple deep Q-learning [4].

There are also cases where reinforcement learning has been applied to the prisoner's dilemma problem. Through simple, repeated prisoner's dilemma experiments, it has been demonstrated that reinforcement learning models can be trained to adopt cooperative strategies rather than just the Nash equilibrium [3]. Additionally, research has been conducted to develop more effective reinforcement learning techniques for scenarios involving games with multiple models [1].

3 Method

Evolution of trust We define the game 'Evolution of trust'. In this game, P players participate in N rounds. In each round, all P players play a total of M Prisoner's Dilemma games with different players. In other words, each player plays M games with another player, resulting in a total of $(P-1)M$ games to determine their score. Based on the scores at the end of the round, $P_{replace}$ players will be replaced. The $P_{replace}$ players with the lowest scores are removed, and $P_{replace}$ new players, characterized by the traits of the highest-scoring players, are introduced. If there are ties in the lowest or highest scores, the selection is made randomly.
Model Each player cannot know the exact traits of the opponent but can access the historical record of past interactions. We design the RL agent's state based on this scenario. *State* is defined as the record against the opponent up to the l-th match. *Action* can be either 'Cooperate' or 'Betray'. The *Reward* is defined as follows: $+2$ for ('Cooperate', 'Cooperate'), $+3$ for ('Betray', 'Cooperate'), -1 for ('Cooperate', 'Betray'), and 0 for ('Betray', 'Betray').

3.1 Q-learning

We apply the fundamental reinforcement learning method, Q-learning. Through this method, we can verify how well the Q-value is estimated in the deep Q-network applied later. Depending on how the state is provided in Q-learning, we create a total of three different models.

In the *Q-learning basic model*, when playing against the i-th player, the state is received as follows: $S_t^i = [D_{t-l+1}^i, \ldots, D_t^i]$ where l is the length of memory and $D_t^i = A_t^{opponent}$. We create two different models based on the history length: one with $l = 2$ referred to as 'RLagent', and one with $l = 4$ referred to as 'smarty'. In the *Q-learning history model*, when playing against the i-th player, the state is received as follows: $S_t^i = [D_{t-l+1}^i, \ldots, D_t^i]$ where l is the length of memory and $D_t = (A_t^{agent}, A_t^{opponent})$. We conduct experiments on three different lengths of history: $l = 3, 4, 5$. In the *Q-learning business model*, when playing against the i-th player, the state is received as follows: $S_t^i = [D_{0,prev}^i, \ldots, D_{l-1,prev}^i, D_{t-l+1,curr}^i, \ldots, D_{t,curr}^i]$ where l is the length of memory, $D_{t,prev} = (A_{t,prev}^{agent}, A_{t,prev}^{opponent})$, and $D_{t,prev}$ is the record of previous round. We conduct experiments only for the case where the history length is $l = 3$.

3.2 Deep Q-Network

The basic structure of the Deep Q-Network follows the reference [4], but we designed our Q-network in a unique way. Since we receive the state as $S_t^i = [D_{t-l+1}^i, \ldots, D_t^i]$, we first encode the action pair D using a one hot vector method, ensuring that information about both cooperate and betray is included in the state. Specifically, if D is ('Cooperate', 'Cooperate'), it is encoded as $[0, 1, 1]$; if D is ('Betray', 'Cooperate'), it is encoded as $[0, 0, 1]$; if D is ('Cooperate', 'Betray'), it is encoded as $[0, 1, 0]$; if D is ('Betray', 'Betray'), it is encoded as $[0, 0, 0]$; and if there is no action, it is encoded as $[1, 0, 0]$. The encoded input is then processed through a convolutional layer to aggregate the information from the action records. This is followed by a fully connected layer, with the output being a (2,1) vector representing the Q-values for each action.

4 Experimental Results

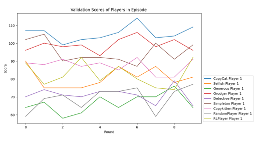
Before starting the experiments, the optimal strategies for dealing with each opponent are as follows: B and C represent "Betray" and "Cooperate," respectively. The "Copycat," "Grudger," and "Simpleton" strategies follow the pattern (C, C, C, C, C, B). The "Selfish," "Generous," and "Random" strategies respond with (B, B, B, B, B, B). The "Detective" strategy shows the pattern (B, B, B, C, C, B). The "Copykitten" strategy alternates responses with (B, C, B, C, B, B).

4.1 Q-learning basic

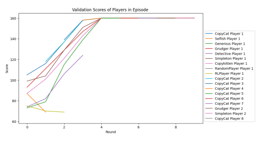
RL Agent and Smarty are trained for 2000 episodes, each episode consists of 1 rounds, and each round consists of 10 games. Scores represent average scores per round in validation mode.

RL Agent The agent successfully identifies strategies against simple opponents like Selfish and Generous but fails to find strategies against more complex opponents, only employing a simple 'Betray' strategy (Tab. 1). This limitation arises because the agent uses only the last 2 actions of the opponent as its state. With only 4 possible states, it is impossible to devise a strategy to beat these more complex opponents. In Fig. 1a, 1b, we tested one-to-many setting with replacement option. Training converges and the agent survives until the 3rd round in Fig. 1b.

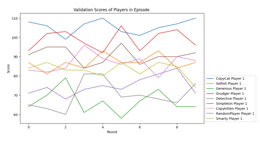
Smarty Despite the history length increased to 4, the agent does not achieve a better score than the RL Agent (Tab. 1). It partially understands Copykitten's behavior, achieving a score of 15, but this is still not optimal. Most opponents base their actions on the agent's previous actions, suggesting that including the agent's own actions in the state representation would lead to more efficient decision-making. Fig. 1c, 1d shows one-to-many setting with replacement option. In the one-to-many setting, training converges, and the agent survives until the 5th round in Fig. 1d.



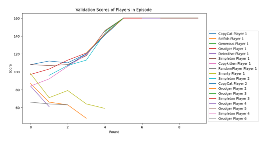
(a) Replace 0



(b) Replace 1



(c) Replace 0



(d) Replace 1

Figure 1: one-to-many results for RL Agent and Smarty

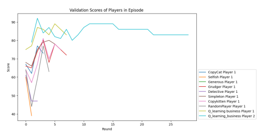


Figure 2: Validation scores of players in a single episode

	RL Agent		Smarty		Q-learning hist 3		Q-learning hist 4	
Opponent	Agt	Opp	Agt	Opp	Agt	Opp	Agt	Opp
Copycat	3	-1	3	-1	20	20	20	20
Selfish	0	0	0	0	0	0	0	0
Generous	30	-10	30	-10	-10	30	-10	30
Grudger	3	-1	3	-1	20	20	20	20
Detective	9	-3	-3	9	19	15	20	12
Simpleton	15	-5	15	-5	20	20	20	20
Copykitten	6	-2	11	-1	25	5	25	5
Random	24	-8	15	-5	15	-5	21	-7

Table 1: Comparison of agent performance in one-on-one experiments across different reinforcement learning setups. 'Agt' stands for Agent scores, and 'Opp' stands for Opponent scores.

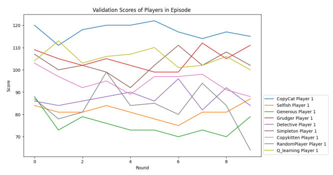
4.2 Q-learning history

Three different models with distinct history lengths of 3, 4, and 5 are trained for two tasks: one-on-one and one-to-many. For the one-on-one task, 1000 episodes are trained, each consists of 1 round and each round consists of 10 games. For the one-to-many task, 200 and 500 episodes are trained, each consists of 10 rounds, and each round consists of 10 games, with testing conducted both with and without the replacement option.

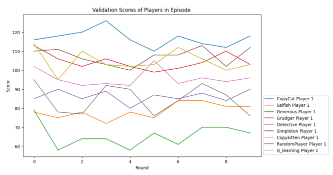
Q learning with history 3 The agent successfully found the sub-optimal strategy for most opponents, including Copykitten, where the agent's pattern was (B, C, B, C, B, C) and Copykitten's was (C, C, C, C, C, . . .) (Tab. 1). However, it failed to find the sub-optimal strategy against Detective due to the short history size, as Detective uses 4 rounds to determine its strategy, making a history of 3 insufficient. Fig. 3a, 3d shows one-to-many setting with epsilon 1. The agent performed lower than CopyCat but better than other agents, surviving until round 10 in Fig. 3d.

Q learning with history 4 The agent successfully found the sub-optimal strategy against Detective, with the agent's pattern being (B, B, B, C, C, C, ...) and Detective's pattern being (C, B, C, C, C, B, ...) (Tab. 1). The agent found sub-optimal strategies in all one-on-one situations. However, it could achieve a higher score by choosing 'Betray' if it knew the round was ending, which it cannot distinguish when the history length plus one is less than the round play number, preventing the discovery of the complete optimal strategy. Fig. 3b, 3e shows one-to-many setting with epsilon 1. The agent performed lower than CopyCat but better than other agents, performing best in high rounds and surviving until round 10 in Fig. 3e.

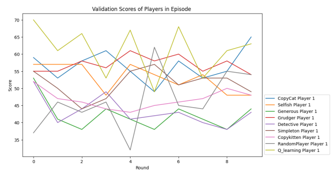
Q learning with history 5 Tab. 2 shows results for one-on-one setting on the Q-learning agent with history sizes 3, 4, and 5. Tab. 2 was tested against all others to analyze the correlation between history size and the number of games in one round. Trained for 5000 episodes, each episode consists of 1 rounds, and each round consists of 6 games. The best performance was observed when the history size minus one equals the number of games. When *history size - 1 < # games*, the agent fails to understand the end-of-round concept and make optimal strategies. The results show that the history size 5 agent performs better only against other opponents, mainly learning to betray at the end of the game. Fig. 3c, 3f shows one-to-many setting. The Q-learning agent with history size 5 performs the best among all agents.



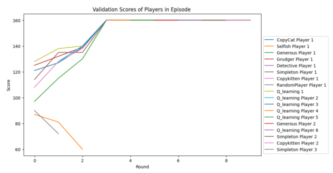
(a) Replace 0



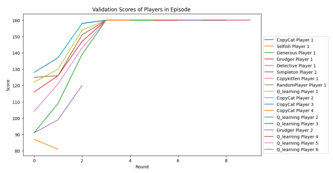
(b) Replace 0



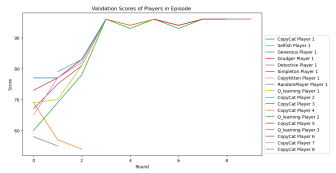
(c) Replace 0



(d) Replace 1



(e) Replace 1



(f) Replace 1

Figure 3: one-to-many results for Q learning

Q-learning business In the one-on-one, the agent with an efficient state representation finds the sub-optimal strategy faster, even with a longer history size, achieving optimal strategies within 500 episodes in environments similar to previous Q-learning setups. Fig. 2 shows one-to-many with replacement, while the history 5 agent maintains balance with many players, the business model exhibits an overwhelming strategy from the start, surviving alone. This significant difference is due to the model's ability to identify the opponent immediately, allowing it to decisively dominate from the second round onwards.

When the history size is 3, there are 85 possible states, and full exploration is confirmed. With a history size of 4, there are 341 possible states, and full exploration is also confirmed. However, when the history size increases to 5, the number of possible states rises to 1365, and full exploration is not achieved. Thus, using a history size of 5 with Q-learning is not efficient for Reinforcement Learning.

4.3 Deep Q-Network

For the one-on-one, 200 episodes are trained, each episode consists of 1 rounds, and each round consists of 6 games. For the one-to-many, 200 episodes are trained, each consists of 10 rounds, and each round consists of 5 games, with testing conducted both with and without the replacement option.

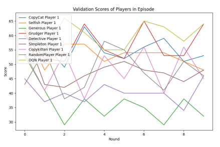
DQN one-on-one In Tab. 2, DQN achieves highest score it can get. Hence, we can assume every solution is completely optimal. When we analysed the Q function, every action was actually optimal.

DQN one-to-many W/o Replace Fig. 4a shows one-to-many setting without replacement. In most rounds, DQN has the highest score. Overall, DQN performs the best among all and Grudger is the runner up. From this result we can guess many players are cooperative and DQN and Grudger is taking advantage of it.

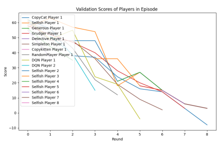
DQN one-to-many W/ Replace and W/ inter-round memory When replacement is available, we made inter-round memory for better performance. In Fig. 4b, DQN performs nice in lower rounds,

	Q-learning history 3		Q-learning history 4		Q-learning history 5		Deep Q-Network	
Opponent	Agent	Opponent	Agent	Opponent	Agent	Opponent	Agent	Opponent
Copycat	12	12	12	12	13	9	13	9
Selfish	0	0	0	0	0	0	0	0
Generous	18	-6	18	-6	18	-6	18	-6
Grudger	12	12	12	12	13	9	13	9
Detective	13	1	13	1	13	1	13	1
Simpleton	12	12	12	12	13	9	13	9
Copykitten	15	3	15	3	16	0	16	0
Random	8	0	9	-3	3	-1	12	-4

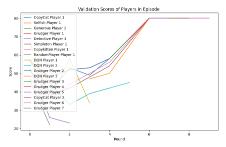
Table 2: Comparison of agent performance in one-on-one experiments across different Q-learning history lengths. Scores represent scores in validation mode.



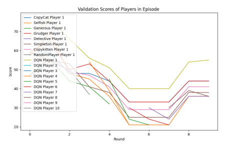
(a) Replace 0



(b) Replace 1, Episode 200 W/ inter-round memory



(c) Replace 1, Episode 600 W/ inter-round memory



(d) Replace 1 W/o inter-round memory

Figure 4: one-to-many results for Deep Q-Network

and gets worse. Group converges to Selfish player and reward gets lower. Unlike our business model's memory this inter-round memory is just a simple queue memory so when it overflows, our state gets shifted. This means our state has previous round's history and current round's history as well. Shifted state makes the model hard to learn correct correlation between the (state, action) pair. However, if we train our model with enough time, model learns some local optimum point like Fig. 4c.

DQN one-to-many W/ Replace and W/o inter-round memory Because of the inter-round memory's problem mentioned above, we tested one-to-many without inter-round memory. Without inter-round memory our model has to find out who is the opponent and how to deal with it in the same round. On the other hand, business model's memory can distinguish the opponent by past history and return optimal action from start of the current round. Hence, this model may not achieve optimal solution in theory but still it can find out the sub-optimal solution that is good enough. In Fig. 4d, our model is the best among all and has no rivals. Also it performs the best in every round.

5 Conclusion

The introduction of reinforcement learning techniques demonstrated that the solution to the one-on-one problem coincides with the optimal solution manually derived by humans. Additionally, in the case of one-to-many, a robust model was designed that possesses effective and feasible solutions in environments that consider the existence and intensity of survival competition, following arbitrarily set player compositions. In the case of one-to-many, our model establishes a stable cooperative relationship with cooperating partners. However, unlike Copycat, it does not cooperate with players who are incapable of forming a steadfast alliance, opting instead for actions that maximize individual gain. Through this approach, we have identified a superior strategy that surpasses the traditionally esteemed Copycat, advancing our understanding of the evolution of trust. The scenario resembles many social and evolutionary biological phenomena. We anticipate that our research will enable us to solve more complex social problems modeled as games by expanding on this approach in the future.

Contribution

Sangmin Lee Subject selection, Preliminary research (Baseline code for the evolution of trust, RL algorithms), Q-Learning and DQN train code, Model design, Result data analysis

Dongwoo Won Subject selection, Preliminary research (Baseline code for the evolution of trust, RL algorithms), Q-Learning and DQN train code, Model design, Result data analysis

Seungwan Kang Subject selection, Preliminary research (Baseline code for the evolution of trust, RL algorithms), Q-Learning and DQN train code, Model design, Result data analysis

References

[1] Milad Aghajohari, Juan Agustin Duque, Tim Cooijmans, and Aaron Courville. LOQA: Learning with opponent q-learning awareness. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FDQF6A1s6M.

[2] Robert Axelrod and William D. Hamilton. The evolution of cooperation. *Science*, 211(4489): 1390–1396, 1981. doi: 10.1126/science.7466396. URL https://www.science.org/doi/abs/10.1126/science.7466396.

[3] Arthur Dolgopolov. Reinforcement learning in a prisoner's dilemma. *Games and Economic Behavior*, 144:84–103, 2024. ISSN 0899-8256. doi: https://doi.org/10.1016/j.geb.2024.01.004. URL https://www.sciencedirect.com/science/article/pii/S0899825624000058.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[5] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.