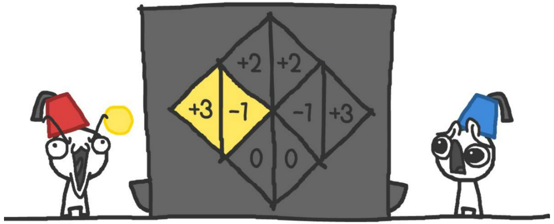
Conquering the Evolutionary Multi-Prisoner's Dilemma with Reinforcement Learning

Team 9

Sangmin Lee, Dongwoo Won, Seungwan Kang

2024.06.10

Preview: The Evolution of Trust



- Playing Prisoner's dilemma with many other players (agents) and get coin according to the result

- Player with many coin stays and player with little coin is replaced by other kind of player

- Game ends when all players are replaced into one kind of player

- There are about 8 types of players. They have their own behavior rule.

Background: Word defined

- **Episode**: Playing game (Prisoner's dilemma) until winner agent occur is called one episode
- **Replacement**: Replacing agents according to their coins and resetting there coins
- **Episode length**: Number of replacement held until episode finishes
- **Round**: Number of game played with each players between each replacement
- **Agent**: Type of player, which determines the behavior (below are 8 simple agents)

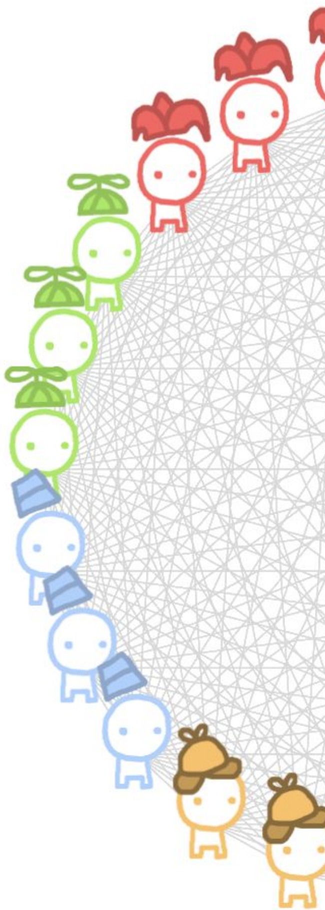
Goal

Find Best Model that on “The Evolution of Trust”

without knowing any other information except opponent’s decision

Experimental Design

1. **One-on-One Experiments:**
   - Conduct one-on-one experiments to evaluate and train the model against simple agents.
2. **One-to-Many Experiments:**
   - After training, extend the experiments to one-to-many scenarios to further validate the model's performance.
3. **Varying History Lengths:**
   - Perform experiments with different history lengths to understand the model's adaptability and effectiveness across various contexts.



Problem Definition

- Goal : Find the best model on game ‘Trust of Evolution’ with Best Q!
- We define the model as:

State: ‘Record against the opponent up to the l-th match’

$$S_t = [D_{t-l+1}, \ldots, D_t]$$

where l is the length of memory, $D_i = (A_{agent}, A_{opponent})$ which is the action pair.

Action: ‘Cooperate’ or ‘Betray’

Reward:

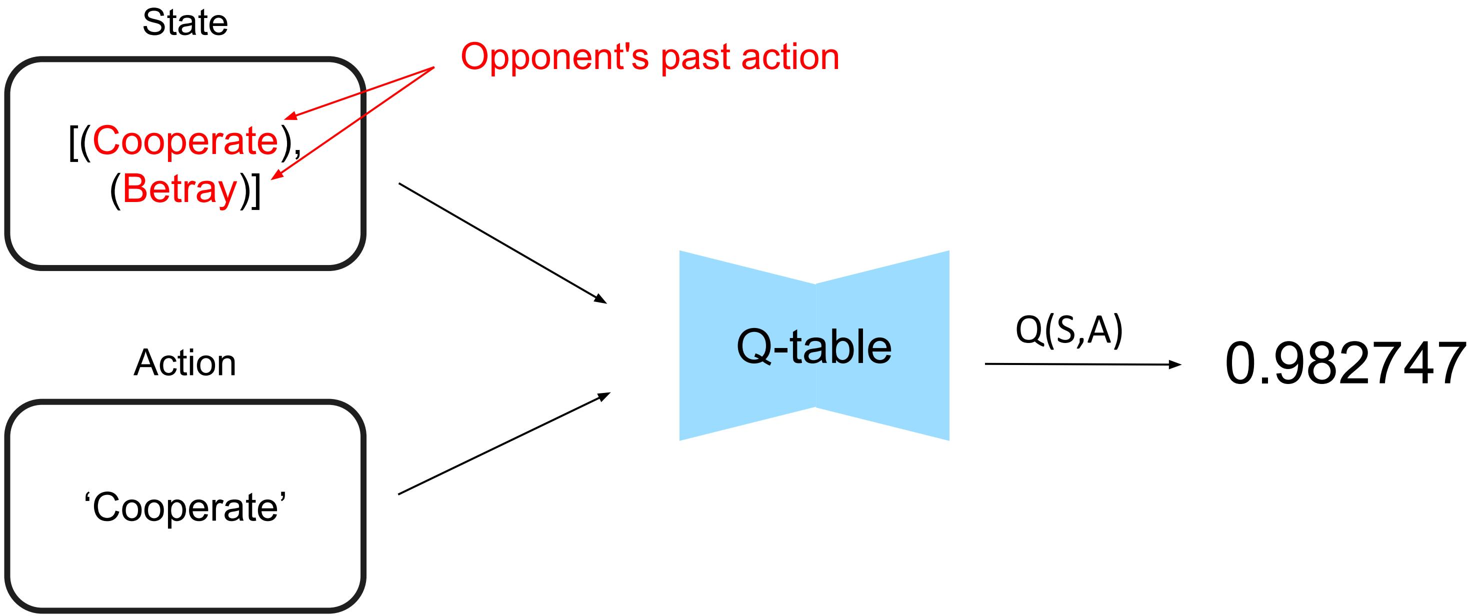
$$\begin{cases} +2 & \text{if } D = (\text{‘Cooperate’, ‘Cooperate’}) \\ +3 & \text{if } D = (\text{‘Betray’, ‘Cooperate’}) \\ -1 & \text{if } D = (\text{‘Cooperate’, ‘Betray’}) \\ 0 & \text{if } D = (\text{‘Betray’, ‘Betray’}) \end{cases}$$

Method

RLagent, Smarty (Q-learning)

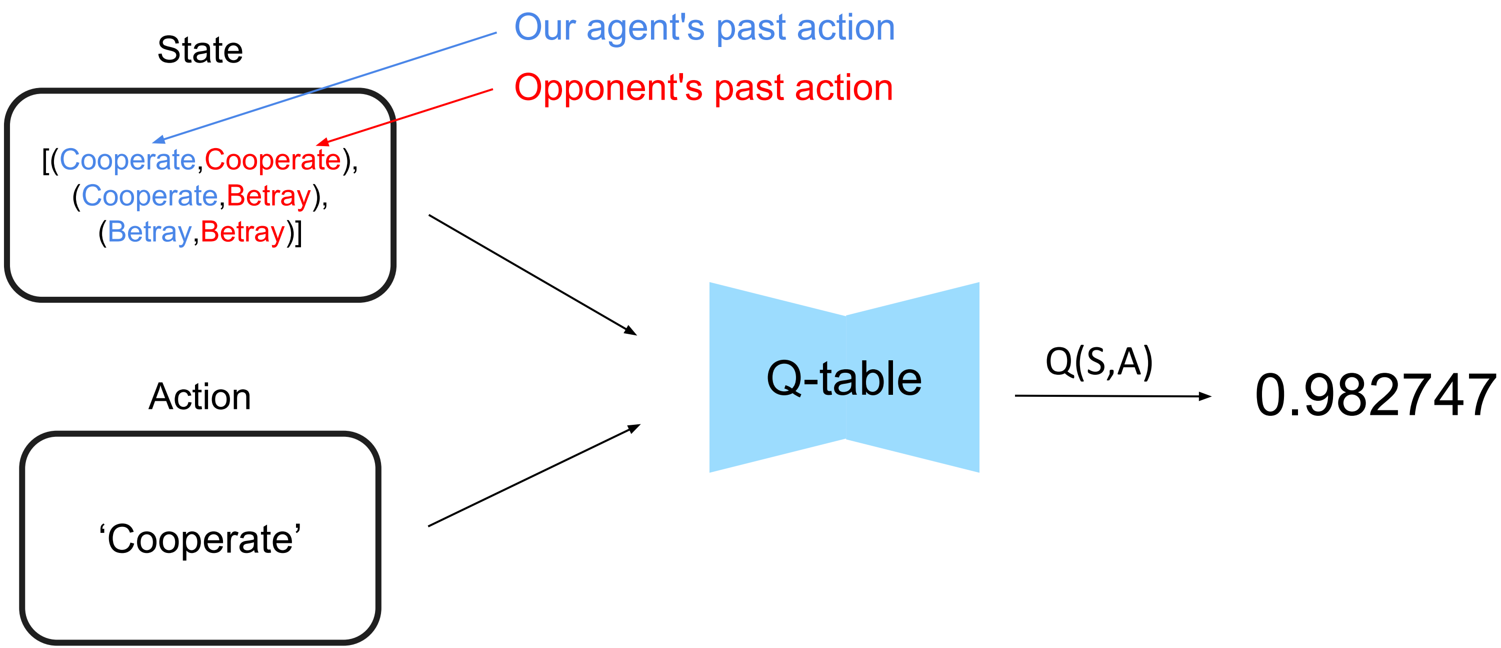
RLagent : (Histroy_length=2, epsilon = 0.1 , gamma = 0.9)

Smarty : (History_length=4, epsilon = 0.08, gamma = 0.95)



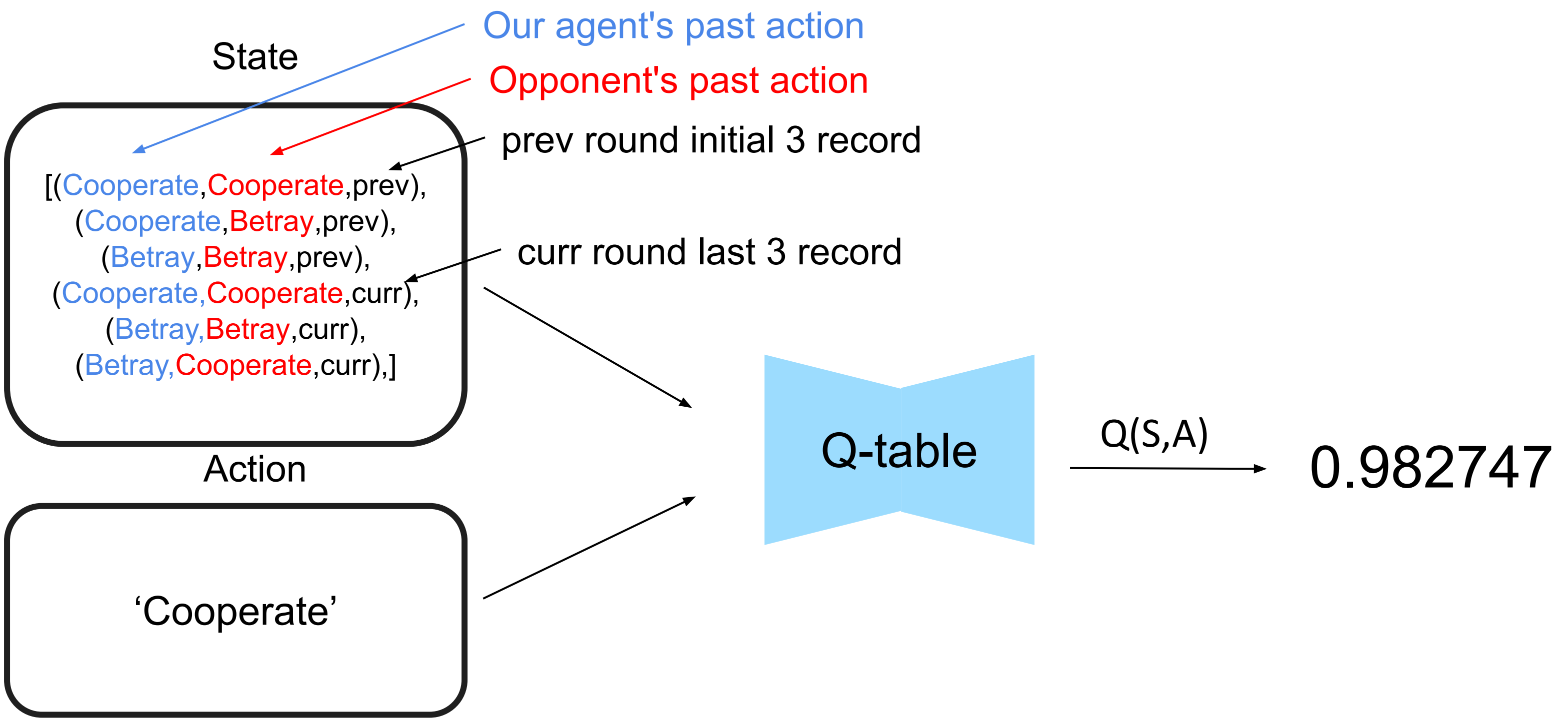
Method

Q-Learning



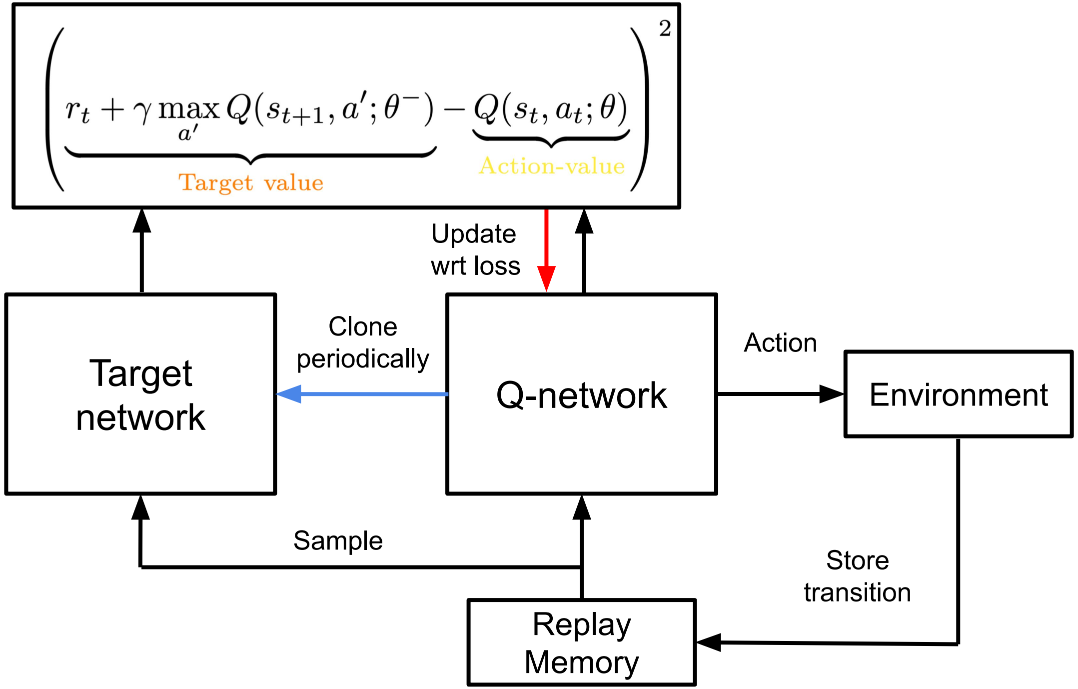
Method

Q-Learning business



Method

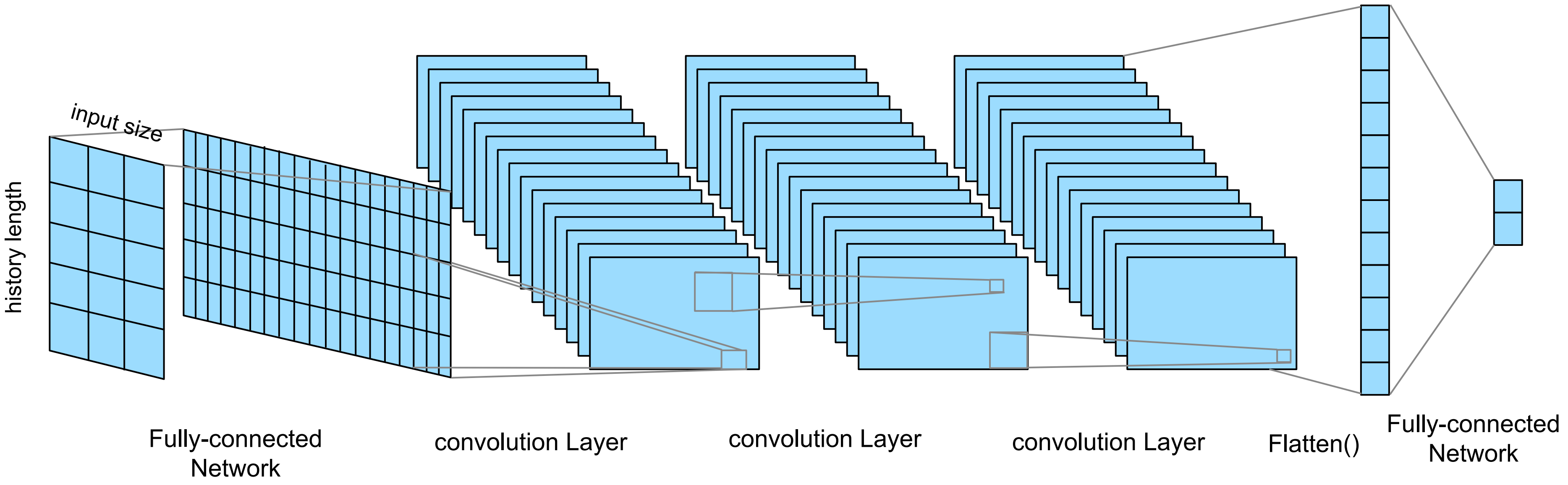
DQN



$$\left(\underbrace{r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)}_{\text{Target value}} - \underbrace{Q(s_t, a_t; \theta)}_{\text{Action-value}}\right)^2$$
Update
wrt loss
Target
network
Clone
periodically
Q-network
Action
Environment
Sample
Store
transition
Replay
Memory

Method

DQN (Q- network)



Results: RL Agent - in 1 vs 1

Table 1: **One-on-One Experiments on RLAgent**, Train for 2000 episodes, each episode consists of 1 rounds, and each round consists of 10 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	3	-1
Selfish	0	0
Generous	30	-10
Grudger	3	-1
Detective	9	-3
Simpleton	15	-5
Copykitten	6	-2
Random	24	-8

Analysis

- The agent successfully identifies strategies against simple opponents like Selfish and Generous.
- Fails to find strategies against more complex opponents.
- Copycat, Copykitten, Detective:
  - the agent only employs a simple 'Betray' strategy.

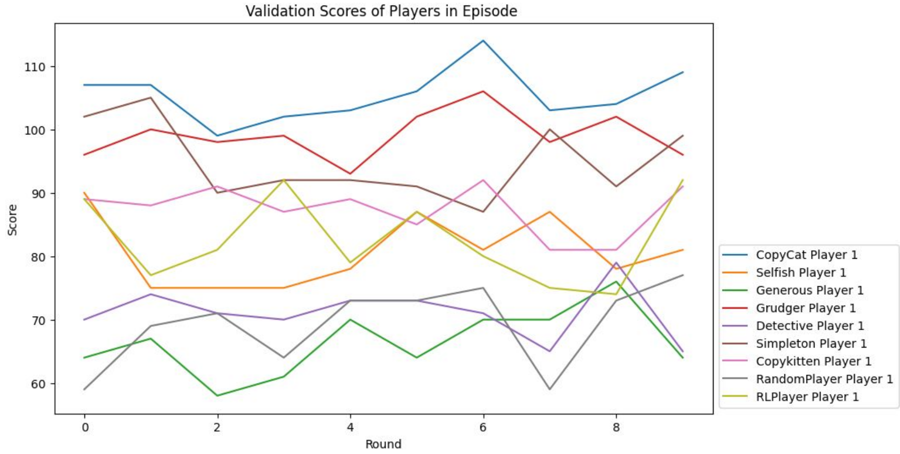
Reason

- 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.

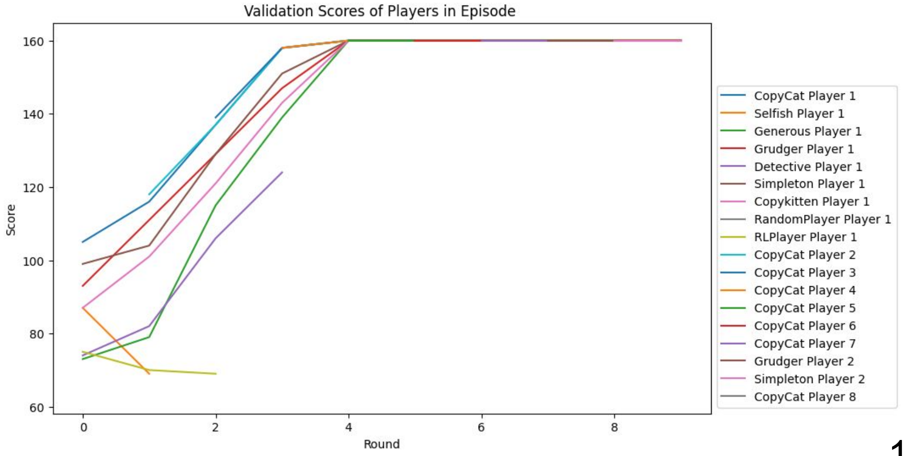
Results: RL Agent - in 1 vs n

- Train Setting: episode: 500, max_len: 10, round: 10
- Result:
    - Training converges.
    - Survived until 3 round

Replace: 0



Replace: 1



Results: Smarty - in 1 vs 1

Table 2: **One-on-One Experiments on Smarty**, Train for 2000 episodes, each episode consists of 1 rounds, and each round consists of 10 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	3	-1
Selfish	0	0
Generous	30	-10
Grudger	3	-1
Detective	-3	9
Simpleton	15	-5
Copykitten	11	-1
Random	15	-5

Analysis

- Despite history length become 4, Agent get not any better score than RLAgent.
- Copykitten:
  - Partially understands Copykitten's behavior, achieving a score of 15, but still not optimal.

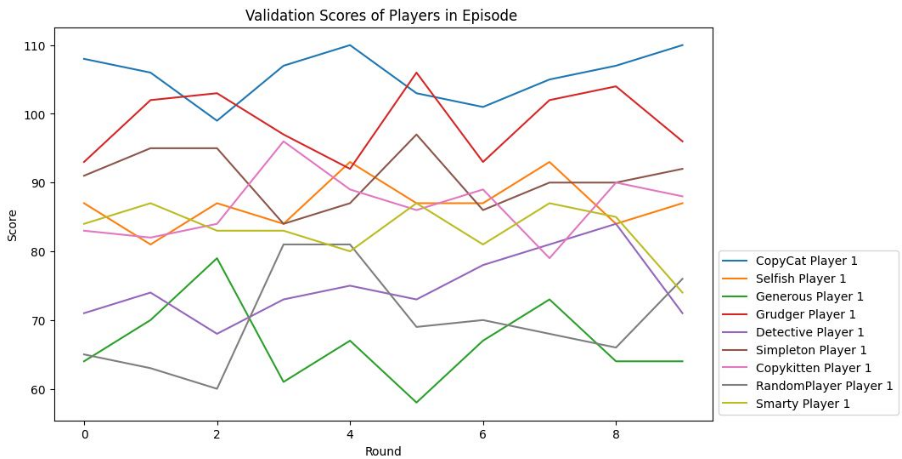
Reason

- 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.

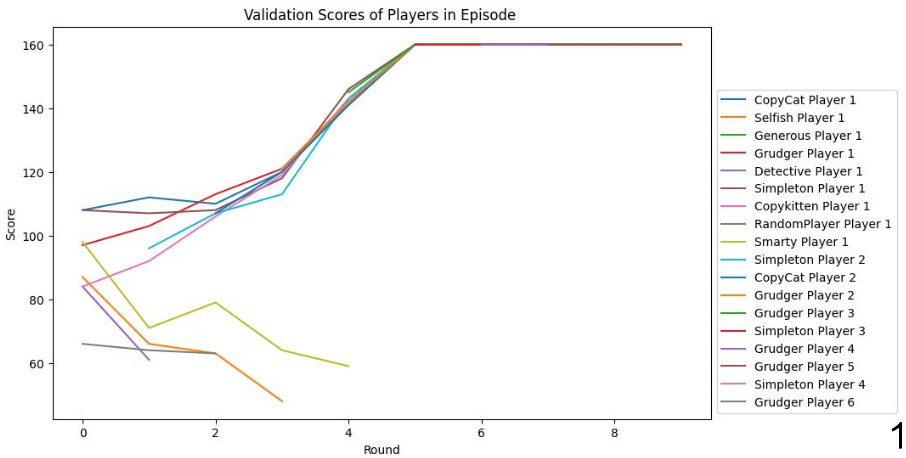
Results: Smarty - in 1 vs n

- Train Setting: episode: 500, max_len: 10, round: 10
- Result:
  - Training converges.
  - Survived until 5 round

Replace: 0



Replace: 1



Results: Q learning with history 3 - in 1 vs 1

Table 3: **One-on-One Experiments on Q-learning with history length 3**,
Train for 1000 episodes, each episode consists of 1 rounds, and each round consists of 10 games. Scores represent average scores per round in validation mode.

Agent	Agent Score	Opponent Score
Copycat	20	20
Selfish	0	0
Generous	-10	30
Grudger	20	20
Detective	19	15
Simpleton	20	20
Copykitten	25	5
Random	15	-5

Analysis

- For most of opponent, successfully found the optimal strategy.
- Copykitten: Successfully found the optimal strategy.
  - agent: (B,C,B,C,B,C)
  - copykitten: (C,C,C,C,C,...)
- Detective: Failed to find the optimal strategy.

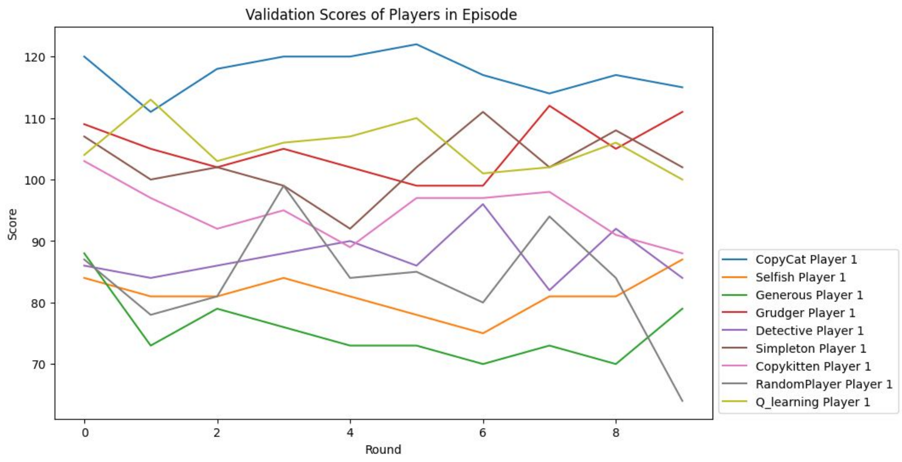
Reason

- Short history size
- Since Detective use 4 round to determine its strategy, history 3 is not enough to beat Detective.

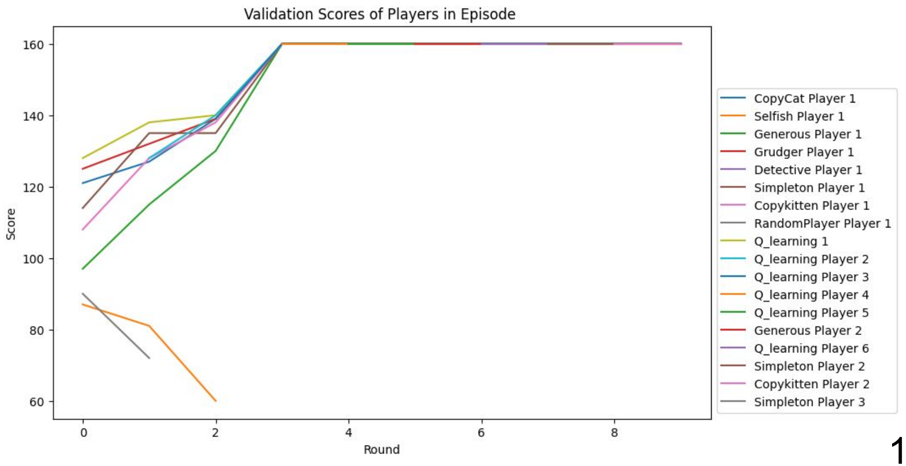
Results: Q learning with history 3 - in 1 vs n

- Train Setting: episode: 200, max_len: 10, round: 10
- Agent Setting: epsilon: 1
- Result:
  - Perform lower than CopyCat but better than other agents
  - Survived until round 10

Replace: 0



Replace: 1



Results: Q learning with history 4 - in 1 vs 1

Table 4: **One-on-One Experiments on Q-learning with history length 4,**
Train for 1000 episodes, each episode consists of 1 rounds, and each round consists of 10 games. Scores represent average scores per round in validation mode.

Agent	Agent Score	Opponent Score
Copycat	20	20
Selfish	0	0
Generous	-10	30
Grudger	20	20
Detective	20	12
Simpleton	20	20
Copykitten	25	5
Random	21	-7

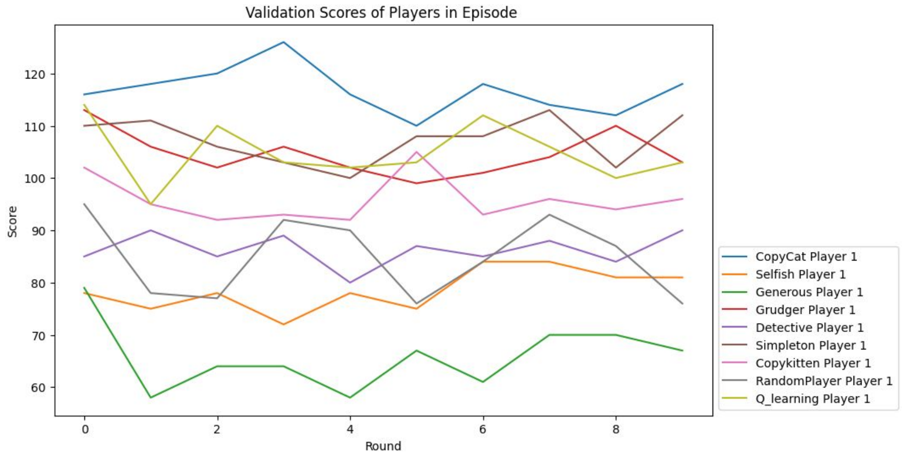
Analysis

- Detective: Successfully found the optimal strategy.
  - agent: (B,B,B,C,C,C, …)
  - Detective: (C,B,C,C,C,B, …)
- Successfully find optimal strategies in all one-on-one situations.
- However, the agent can achieve a higher score by choosing 'Betray' if it knows the round is ending.
- The agent cannot distinguish this in situations where history length + 1 < round play number, preventing the discovery of the complete optimal strategy.

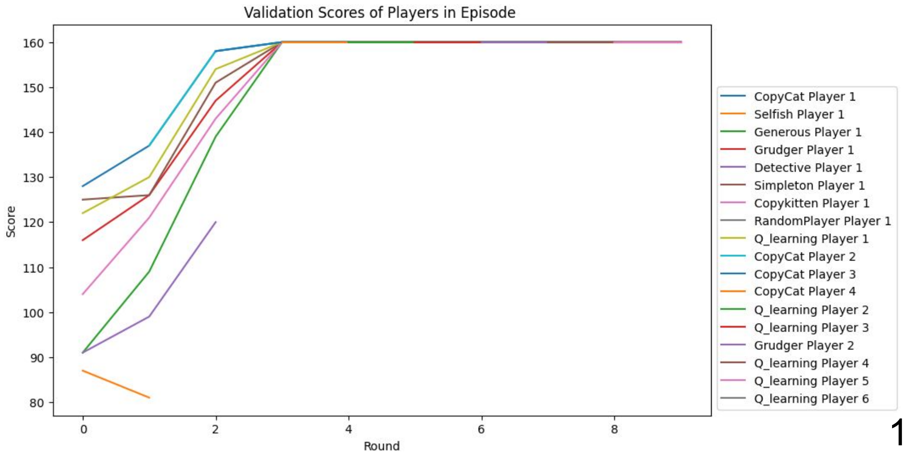
Results: Q learning with history 4 - in 1 vs n

- Train Setting: episode: 500, max_len: 10, round: 10
- Agent Setting: epsilon: 1
- Result:
  - Perform lower than CopyCat but better than other agents
  - In high rounds, it performs best
  - Survived until round 10

Replace: 0



Replace: 1



Results: Q learning with history 5 - in 1 vs 1

- Tested 1 vs 1 for all other players using Q-learning agent with history 3,4,5
- Result:
  - History size 5 only performs better on Copycat, Grudger, Simpleton
  - It mainly learns to betray in the end of the game

Table 5: **One-on-One Experiments on Q-learning with history length 3**,
Train for 5000 episodes, each episode consists of 1 rounds, and each round consists of 6 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	12	12
Selfish	0	0
Generous	18	-6
Grudger	12	12
Detective	13	1
Simpleton	12	12
Copykitten	15	3
Random	8	0

Table 6: **One-on-One Experiments on Q-learning with history length 4**,
Train for 5000 episodes, each episode consists of 1 rounds, and each round consists of 6 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	12	12
Selfish	0	0
Generous	18	-6
Grudger	12	12
Detective	13	1
Simpleton	12	12
Copykitten	15	3
Random	9	-3

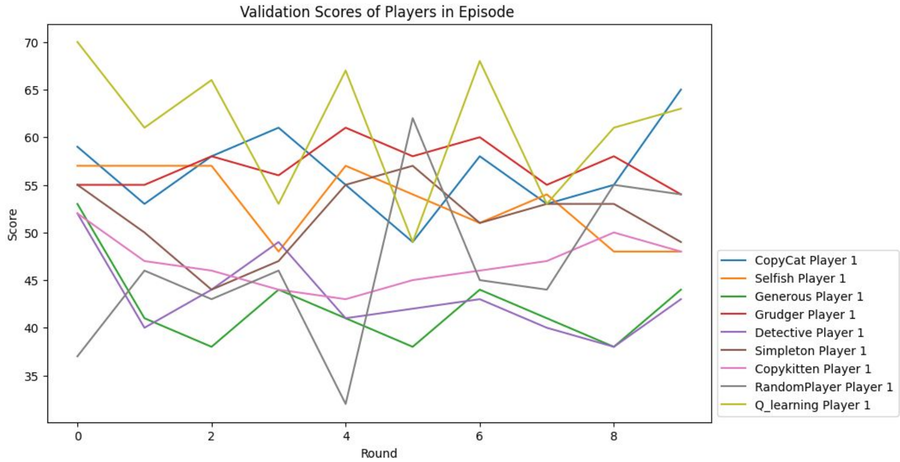
Table 7: **One-on-One Experiments on Q-learning with history length 5**,
Train for 5000 episodes, each episode consists of 1 rounds, and each round consists of 6 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	13	9
Selfish	0	0
Generous	18	-6
Grudger	13	9
Detective	13	1
Simpleton	13	9
Copykitten	16	0
Random	3	-1

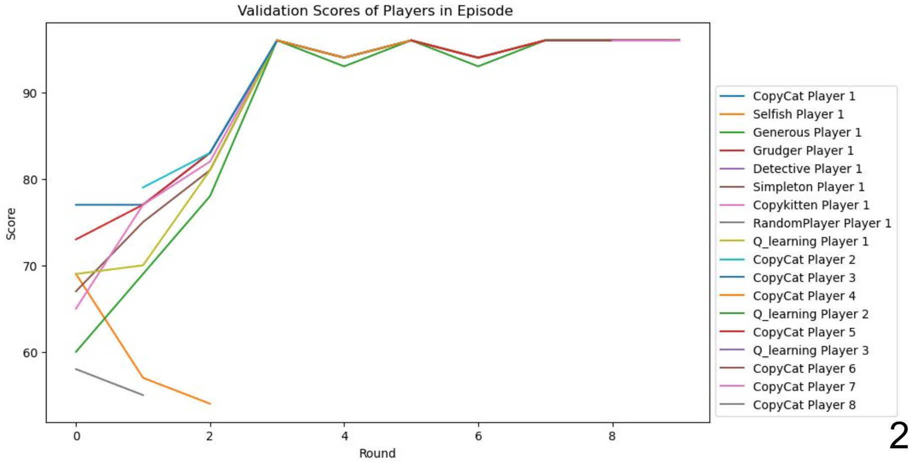
Results: Q learning with history 5 - in 1 vs n

- Train Setting: episode: 500, max_len: 10, round: 6
- Agent Setting: epsilon: 1
- Result:
    - Q-learning best performs among all agents

Replace: 0



Replace: 1



Results: Q learning Business - in 1 vs 1

Table 8: **One-on-One Experiments on Q-learning business with history length 6**, Train for 500 episodes, each episode consists of 1 rounds, and each round consists of 10 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	20	20
Selfish	0	0
Generous	30	-10
Grudger	20	20
Detective	20	12
Simpleton	20	20
Copykitten	25	5
Random	21	-7

Analysis

- Efficient State Proposal:
  - By proposing an efficient state representation, the agent finds the optimal strategy faster, even with a longer history size.
  - The proposed state representation allows the agent to find the optimal strategy within 500 episodes, even in environments similar to previous Q-learning setups.

Results: Problem of Q learning

```
(empd) (base) root@cs580-3:~/EMPD# python pkl.py
q_learning_history3:  85
```

```
(empd) (base) root@cs580-3:~/EMPD# python pkl.py
q_learning_history4:  341
```

```
(empd) (base) root@cs580-3:~/EMPD# python pkl.py
q_learning_history5:  1333
```

- When history size is 3:
  - All possible states: 85
  - Full exploration confirmed.
- When history size is 4:
  - All possible states: 341
  - Full exploration confirmed.
- When history size is 5:
  - All possible states: 1365
  - Full exploration not achieved.
  - Using history size 5 with Q-learning is not efficient for Reinforcement Learning.

Results: DQN with history 5 - in 1 vs 1

Table 9: **One-on-One Experiments on DQN with history length 5**,
Train for 200 episodes, each episode consists of 1 rounds, and each round consists of 5 games. Scores represent scores in validation mode.

Agent	Agent Score	Opponent Score
Copycat	11	7
Selfish	0	0
Generous	15	-5
Grudger	11	7
Detective	11	-1
Simpleton	11	7
Copykitten	13	1
Random	12	-4

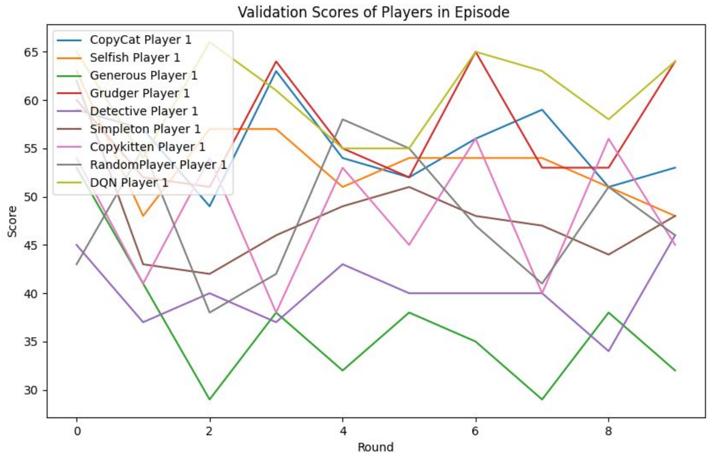
Analysis

- Every solution is complete optimal!
  - Copycat: (C,C,C,C,B)
  - Selfish: (B,B,B,B,B)
  - Generous: (B,B,B,B,B)
  - Grudger: (C,C,C,C,B)
  - Detective: (B,B,B,C,B)
  - Simpleton: (C,C,C,C,B)
  - Copykitten: (B,C,B,C,B)
  - Random: (B,B,B,B,B)

Results: DQN with history 5 - in 1 vs n: W/o Replace

- Train Setting: episode: 200, max_len: 10, round: 5
- Agent Setting: epsilon: 0.5
- Result:
    - Best among all (Grudger is the runner up)
    - In most rounds, it performs the best

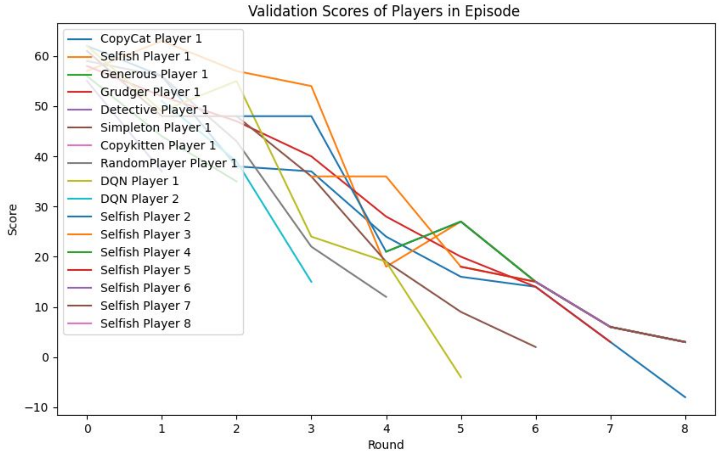
Replace: 0



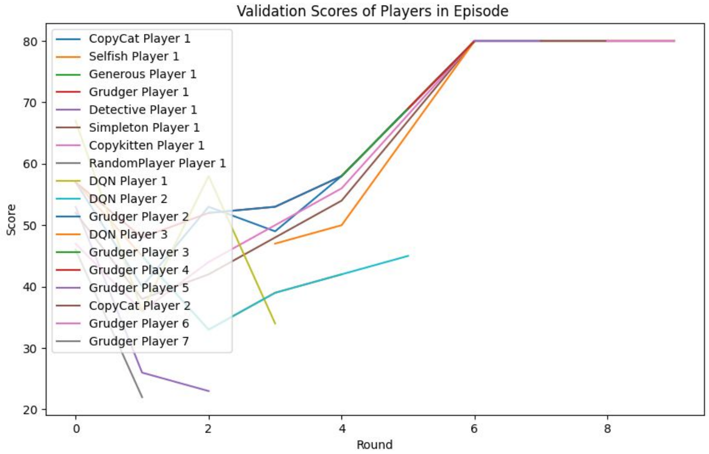
Results: DQN with history 5 - in 1 vs n: W/ Replace

- Train Setting: episode: 200, max_len: 10, round: 5
- Agent Setting: epsilon: 0.5
- Result:
  - Performs nice in lower rounds, and gets worse
  - Converges to Selfish player and reward gets lower

Replace: 1(W/ inter-round memory)



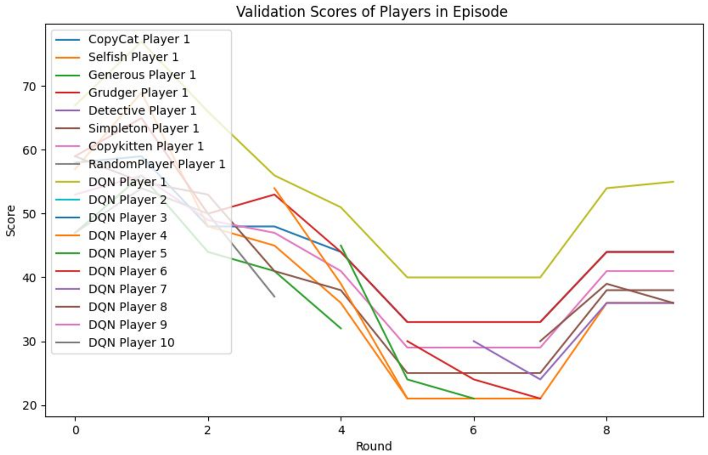
After 600 Episode



Results: DQN with history 5 - in 1 vs n: W/ Replace

- Train Setting: episode: 200, max_len: 10, round: 5
- Agent Setting: epsilon: 0.5
- Result:
    - Best among all, no rivals
    - Every round, it performs the best

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



Conclusion

- **Q-Learning Method:**
  - The Q-learning method successfully learned the correct solution for our problem.
  - However, it is not yet efficient enough, especially as the history size increases.
- **Impact of Q-Learning History Compared to Round Number:**
  - The effectiveness of the Q-learning method depends significantly on the relationship between the history size and the round number.
  - A history size that is greater than or equal to the round number is crucial for finding the optimal strategy.
- **New State Definition:**
  - Introducing a new definition for the state can enhance the performance of the learning algorithm.
- **DQN (Deep Q-Network):**
  - The DQN method requires fewer episodes to learn and provides more accurate solutions.
  - However, the performance is highly dependent on how the state, particularly the memory (history), is defined.
  - An incorrect state definition can prevent the DQN from finding the optimal solution.

Reference

- Nicky Cast, ‘The evolution of trust’ https://ncase.me/trust/
- Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013). https://arxiv.org/abs/1312.5602

Thank you