REINFORCMENT LEARNING FINAL PROJECT

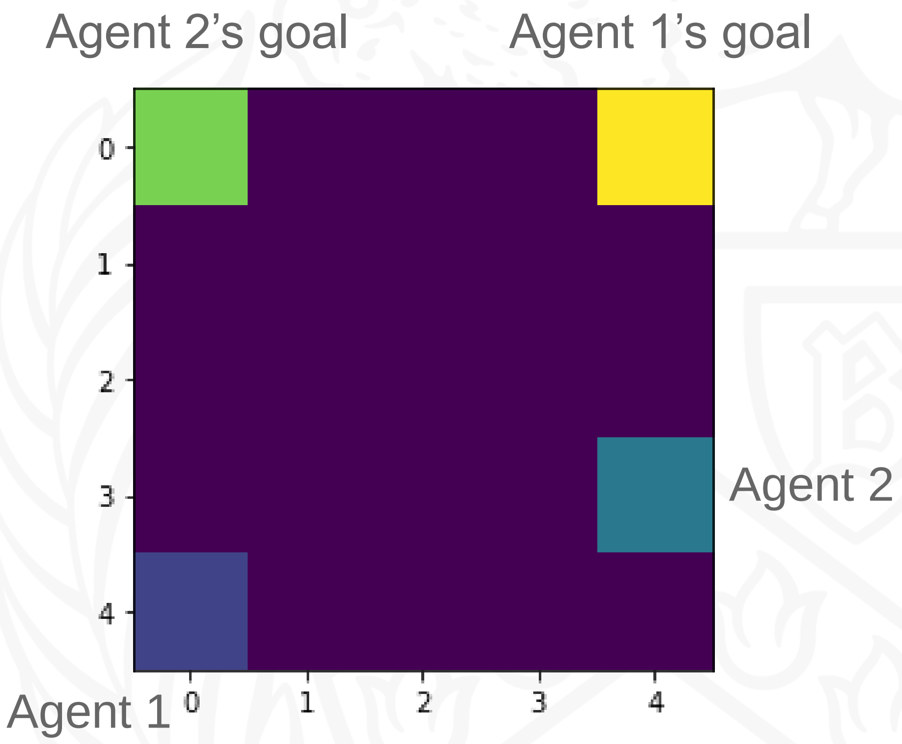
Team 17

Sol Jang, Sungjoon Park

University at Buffalo The State University of New York

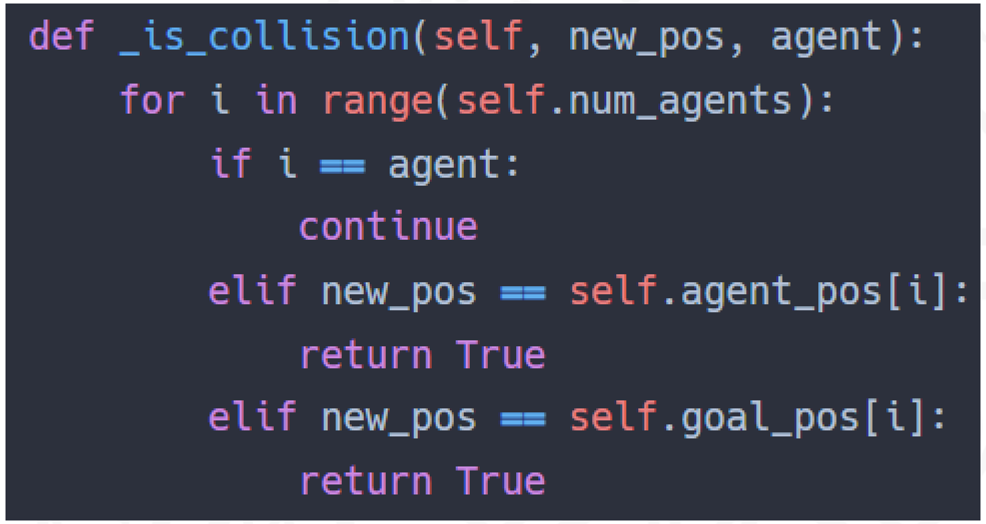
Multi-agent RL

- Objective
  - Implement a decentralized, competitive basic Multi-agent RL (two agents)
  - Find the problems generated in the Multi-agent RL abd resolve them
  - Expand the environment to a more complicated one
- GridWorld Environment
  - Referred to the env discussed in the class
  - Methods: reset(), step(), render()
- Q-learning Agents (referred to spark55's Assignment 1)

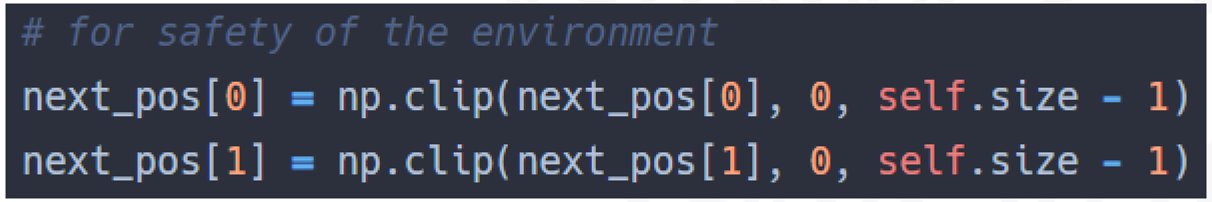


GridWorld Environment with two agents

- Starting points and Goal positions
  - It is arranged diagonally and designed so that each agent encounters each other.
- Collision
  - Attempts to move to a location occupied by another agent are impossible.
- Order
  - For every step, the first agent moves first.
- AI safety
  - Agents can't get out of the GridWorld.



```
def _is_collision(self, new_pos, agent):
    for i in range(self.num_agents):
        if i == agent:
            continue
        elif new_pos == self.agent_pos[i]:
            return True
        elif new_pos == self.goal_pos[i]:
            return True
```

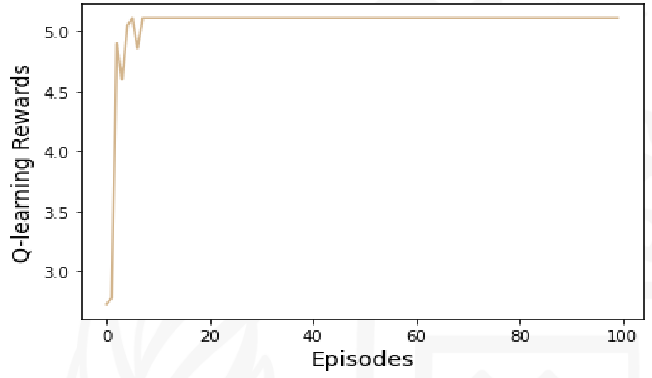


```
# for safety of the environment
next_pos[0] = np.clip(next_pos[0], 0, self.size - 1)
next_pos[1] = np.clip(next_pos[1], 0, self.size - 1)
```

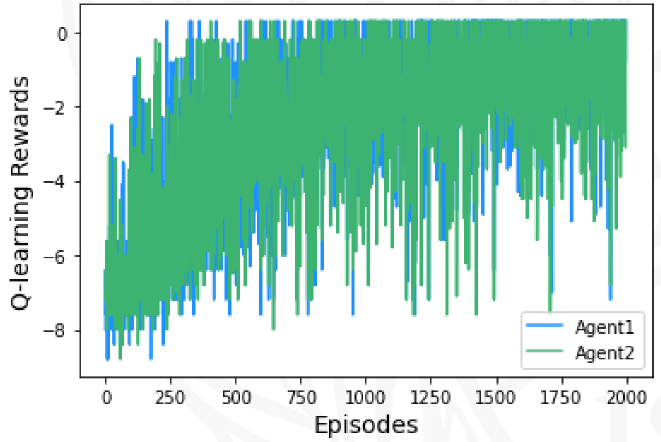
Two agents case and issues

- Trained two Q-learning agents for 2000 episodes
  - Methods: get_action(), update_sample()
- Results
  - Too volatile compared to single-agent cases
  - Outcomes are not desirable for many trials with learnt greedy policy. Please see the results in the corresponding Jupyter Notebook cell.
- Does the order matter?

trained single agent's outcome



trained two agents' outcome



Resolve the issues found in two agents case

- Introduce the random priority.

```python
def step(self, actions):
    # Instantiate dones.
    self._timestep += 1
    old_pos = self.agent_pos.copy()
    # Instantiate rewards.
    rewards = [0] * self.num_agents
    # the next states and the rewards for each player

    for agent in range(self.num_agents):
```

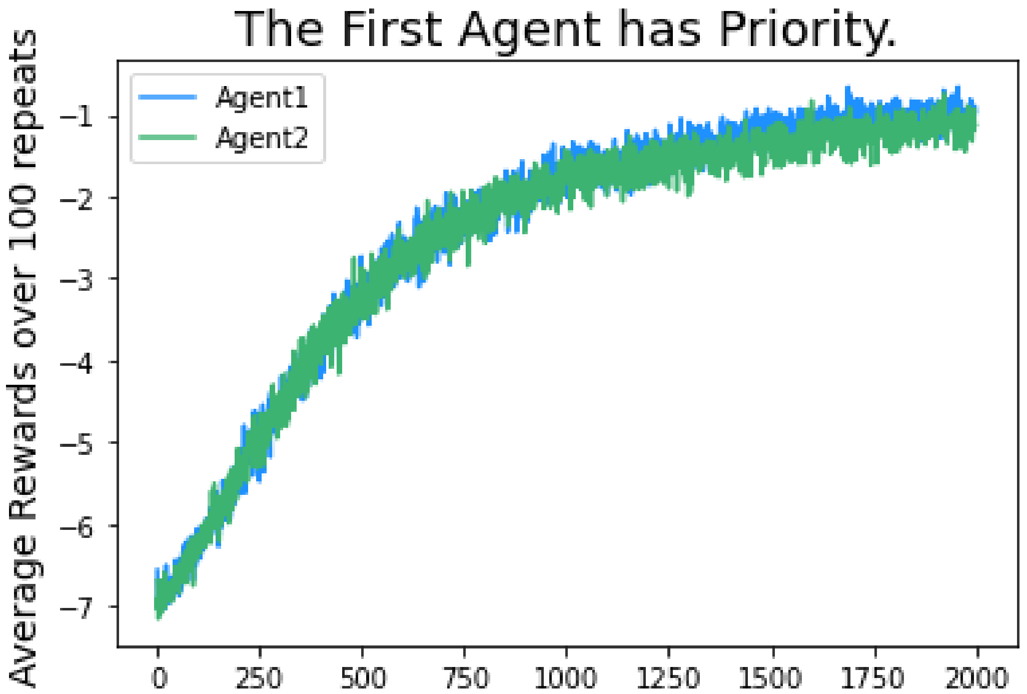
```python
def step(self, actions, temp_random):
    # Instantiate dones.
    self._timestep += 1
    old_pos = self.agent_pos.copy()
    # Instantiate rewards.
    rewards = [0] * self.num_agents
    # the next states and the rewards for each player

    if temp_random == 1:
        order = [0, 1]
    else:
        order = [1, 0]
    for agent in order:
```

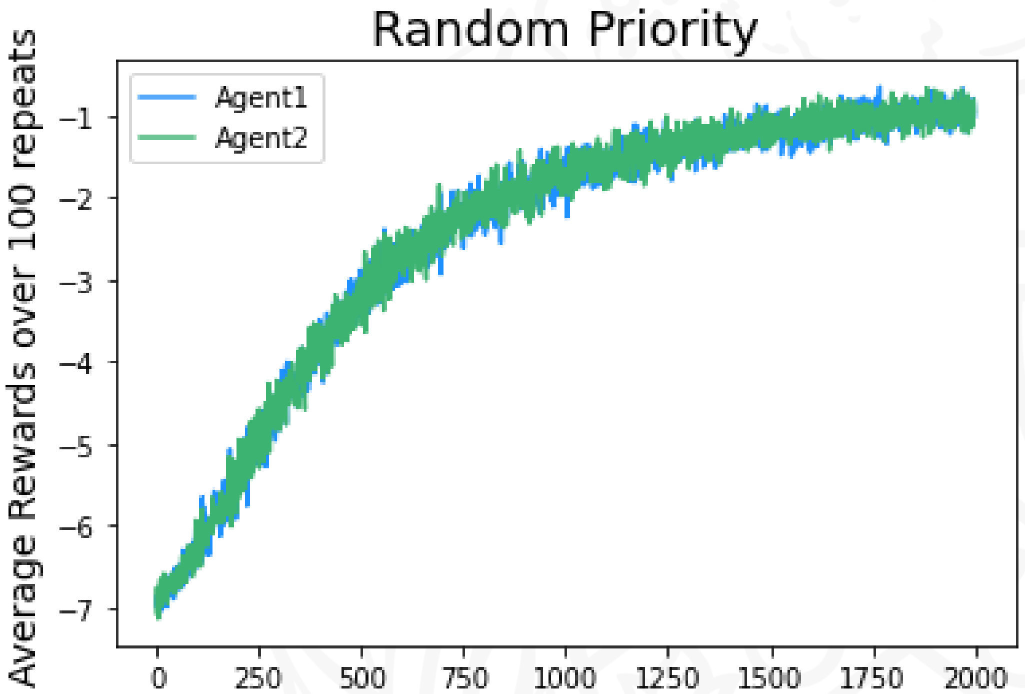
```python
temp_random = np.random.choice(2)
_, rewards, dones, _ = env2.step(actions, temp_random)
```

- The first agent had no advantage over another just because it had priority.
- However, the random priority did the trick so that the agents always show desirable outcomes when using learnt greedy policy. ← Regardless of the ownership of the priority

University at Buffalo The State University of New York

Resolve the issues found in two agents case





The structural difference caused by priority is 0.08.

The structural difference without priority is -0.002.

Expansion to the case with agents more than two

```
config3 = {
    'NUM_AGENTS': 4,
    'GRID_SIZE': 13
}
```

Set those options so that the interval between adjacent agents is an integer.
Interval: $(size - 1) / (num_{agetns} - 1)$

```
if temp_random == 0:
    order = [0, 1, 2, 3]
elif temp_random == 1:
    order = [0, 1, 3, 2]
elif temp_random == 2:
    order = [0, 2, 1, 3]
elif temp_random == 3:
    order = [0, 2, 3, 1]
elif temp_random == 4:
    order = [0, 3, 1, 2]
```

Total 24 random cases
$24 = 4!$

```
elif temp_random == 22:
    order = [3, 2, 0, 1]
else:
    order = [3, 2, 1, 0]
```

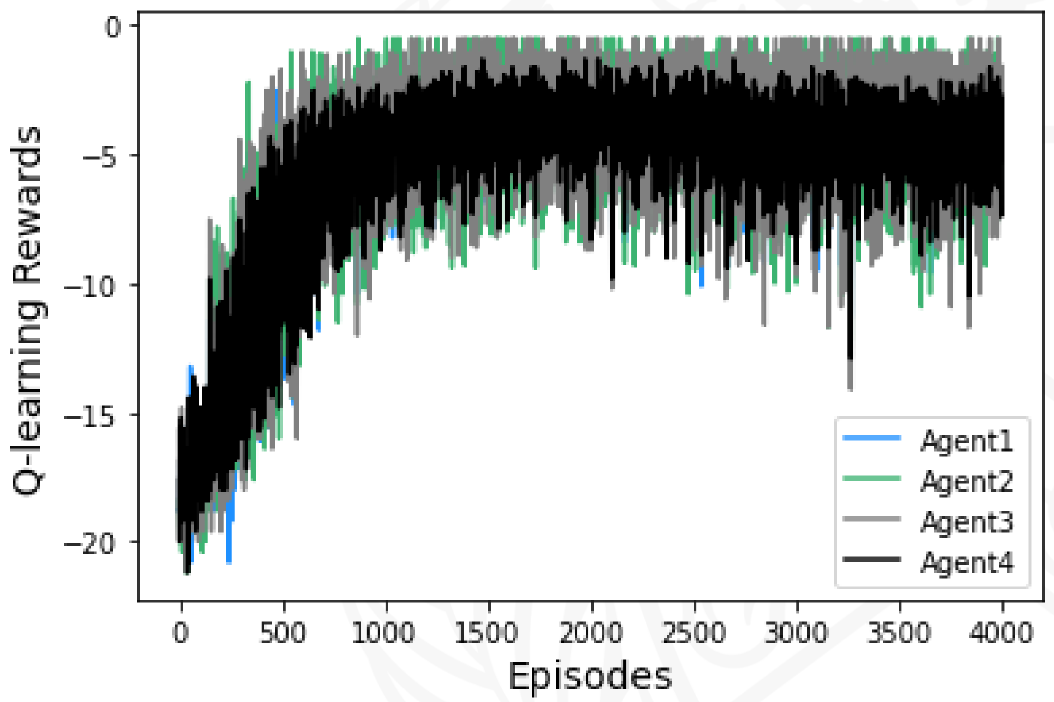
```
temp_random = np.random.choice(24)
_, rewards, dones, _ = env3.step(actions, temp_random)
```

Issues found in the four agents case

MUCH LONGER TRAINING TIME (EPISODES)

- The two-agent case takes about 2.5 seconds to be trained during 2000 episodes.
- The four-agent case takes about 20.7 seconds to be trained during 4000 episodes. However, its agents do not succeed in reaching their goals when using learnt greedy policy. On the other hand, the two-agent case is successful with the training results via 2000 episodes.

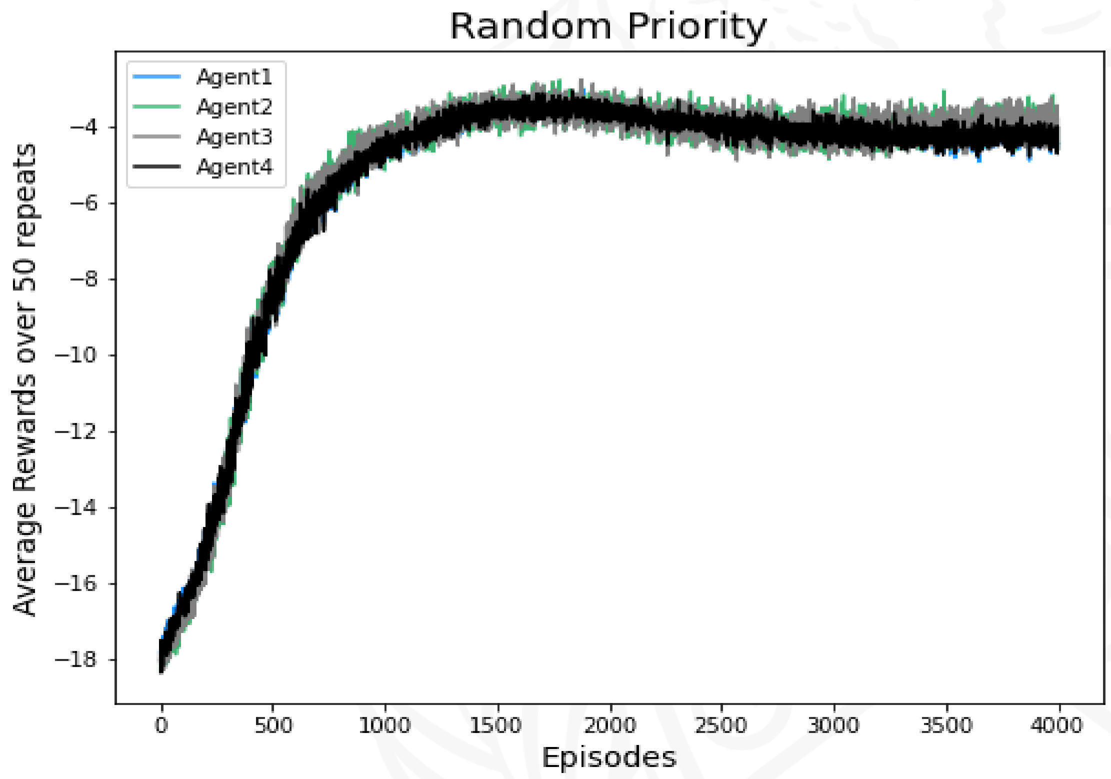
VOLATILE REWARDS & UNDESIRABLE RESULTS



Issues found in the four agents case contd.

MUCH LONGER TRAINING TIME (EPISODES)

- For the four-agent case to be successful, it is required to train the agents for 15000 episodes, which consumes about 69.6 seconds.
- Results
  - One of the four agents can reach the goal but not within the optimal number of steps.
  - Some agents roam around their starting points.



Limitations and Future Works

LIMITATIONS

- Too simple Multi-agent RL
  - Agents were not able to observe other agents' locations and orientations.
  - Centralized and cooperative RL would bring about better results.
- Randomized priority caused a complicated trajectories, which is responsible for the large volatility in rewards. Instead, using a centralized social planner who controls all agents' decisions just like traffic lights will be desirable.

FUTURE WORKS

- First, implement a centralized and cooperative multi-agent RL.
- After successful implementation, try more complicated multi-agent RL such as Hide and Seek.

Contribution

Team Member	Project Part	Contribution (%)
Sol Jang	- Searched for Multi-agent RL materials. - Implemented two-agent RL environments. - Made the presentation.	50%
Sungjoon Park	- Searched for Multi-agent RL materials. - Implemented Q-learning agents. - Created and organized figures.	50%

Reference

Multi-agent Reinforcement Learning (MARL) class lecture note.
From Single-Agent to Multi-Agent to Multi-Agent Reinforcement Learning: Foundational Concepts and Methods, Instituto Superior T´ecnico, Lisbon, Portugal, 2005.

A. Vereshchaka. (2022). spring22_rl_marl_demo [Jupyter Notebook file].
B. S. Park. (2022). Assignment 1 (Checkpoint + Part 2) Sungjoon Park [Jupyter Notebook file].

THANK YOU