Miguel Arpa Robig - 1668833
https://github.com/Miguel231/RL-PRA2

Practical Activity 2

Part 1: Prisoner’s Dilemma

1. Open the iql.py file and implement the missing parts of the IQL class.
    - You will need to complete the act() and update() methods that implement the epsilon-greedy action selection and update the Q-function for the given experience.
    - Also, feel free to modify the schedule epsilon as a hyperparameter. hyperparameters function that

1. act()

Each agent follows an e-greedy policy, independently selecting actions based on its own Q-table. Given an observation, the agent:

- Selects a random action with probability e (exploration).
- Selects the actions that maximizes its Q-value with probability 1-e (exploitation).

2. learn()

For each agent i, the Q-value is updated using the Q-learning update rule.

3. schedule_hyperparameters()

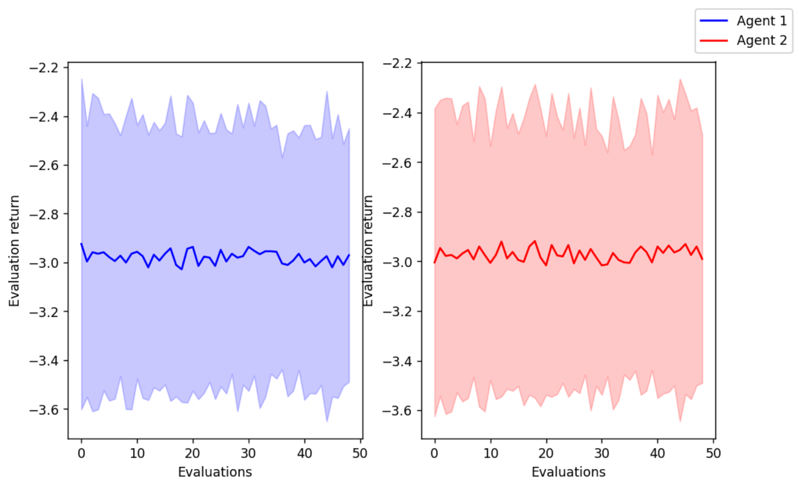
This part remains equal, no changes added.

2. Once you have implemented these methods, you can run the training and evaluation loop by executing the following command: 1 python train_iql.py
    - This will train the IQL agents in the Prisoners’ Dilemma matrix game and periodically evaluate their performance.

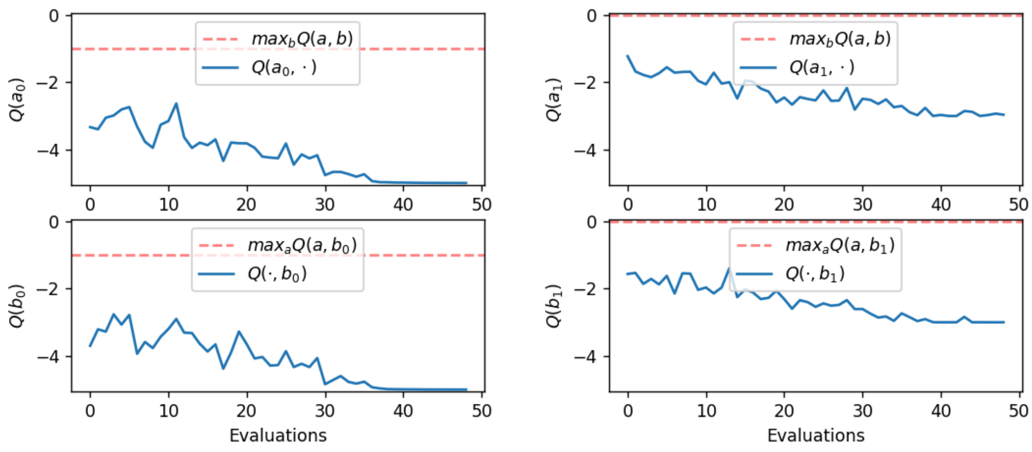
The agents were trained for 20000 episodes, with evaluations every 400 episodes.

3. Once the training process concludes, a standard evaluation is performed, and the results are presented as plots of the learned Q-values for each agent.
    - Include these plots in the report and comment on the training dynamics (e.g., convergence behavior).

IQL:



- Both agents converge to approximately -3.0 return. This corresponds to the (Defect, Defect)
- The mean return stabilizes early while there is variance.



- The plots show that the value for cooperation actions tends to decay to -5, while the value for the defecting action stabilizes around -3.
- Maximum Q-values (red lines) are -1 and 0 respectively. The agents cannot be trusted to cooperate.
- Clear preference emerges for defection over cooperation

- Determine whether any solution concept can be identified in the resulting policies. Discuss and justify your answer.

The learned policies converge to the pure strategy Nash Equilibrium of the Prisoner's Dilemma Defect, Defect(-3,-3).

- Low Q-values for cooperation (= -5)
- Stable equilibrium with no incentive to deviate

IQL agents successfully learn the Nash Equilibrium (Defect, Defect), demonstrating the classic Prisoner's Dilemma outcome where individual rationality leads to collective suboptimality. They prioritise individual security.

4. Implement the Central Q-Learning (CQL) algorithm in a file named cql.py. You may reuse components from the IQL implementation or develop the algorithm entirely from scratch. As in the previous exercises:
In CQL, a single agent learns a Q-function over observations and joint actions. CQL considers the actions of all agents and optimizes the joint reward.

1. __init__()

It receives the number of agents, their action spaces and the hyperparameters.

2. compute_joint_actions()

This new function computes the Cartesian product of the individual agents' action spaces, returning the full set of possible joint actions. In this env, this results in four joint actions.

3. act()

This function implements an e-greedy policy over joint actions, It works equal as the act() function from iql algorithm.

4. learn()

It updates the centralized Q-function using the standard Q-learning update rule applied to joint states and joint actions. The reward sign used is the sum of individual rewards, maximizing the total return.

5. schedule_hyperparameters()

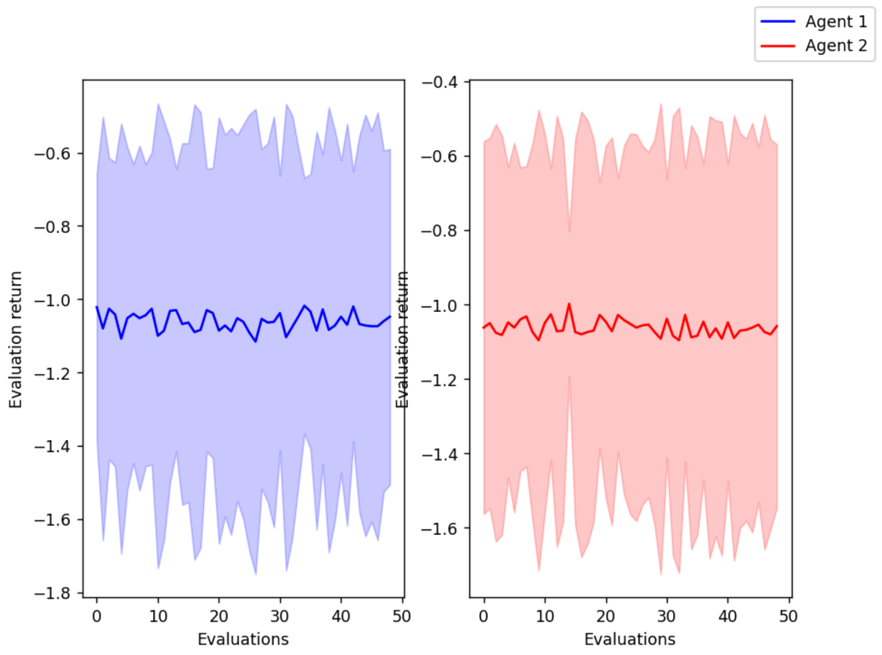
This part remains equal, no changes added.

5. Train the CQL algorithm on the Prisoner's Dilemma matrix game.

The agents were also trained for 20000 episodes, with evaluations every 400 episodes.

6. Perform a standard evaluation and include in the report the results, an analysis of the training dynamics (e.g., convergence behavior), and a discussion of whether any solution concept can be identified in the resulting policies.

CQL:



Unlike IQL, both agents converge approximately to -1.6. This corresponds to the (Cooperate, Cooperate) outcome. The variance suggests that the underlying policy may still involve exploration instances where an agent defects.

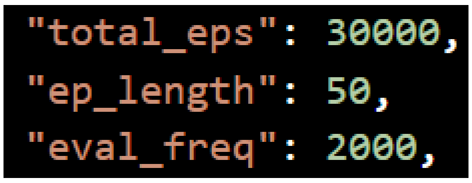
The solution concept identified in this case is the Social Optimum (Pareto Optimal) outcome, Cooperative Equilibrium. This is because CQL is designed to improve coordination in multi-agent env. The agents avoided the dominant strategy of defection in favor of the higher-rewarding cooperative strategy.

Part 2: Level-based Foraging

1. Train both an IQL and a CQL model to solve the LBF environment under the following two configurations:

- (a) Environment Foraging-5x5-2p-1f-v3
- (b) Environment Foraging-5x5-2p-1f-coop-v3

All the models were run following the same number of total episode, episode length:



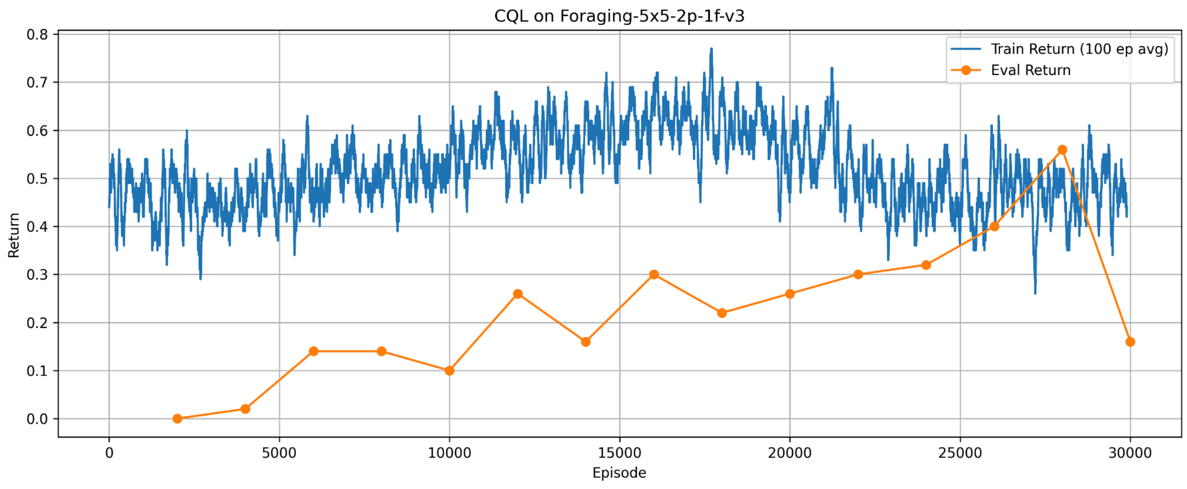
```
"total_eps": 30000,
"ep_length": 50,
"eval_freq": 2000,
```

2. Log the training data and plot the mean episode returns.

- CQL_Foraging_5x5_2p_1f_v3

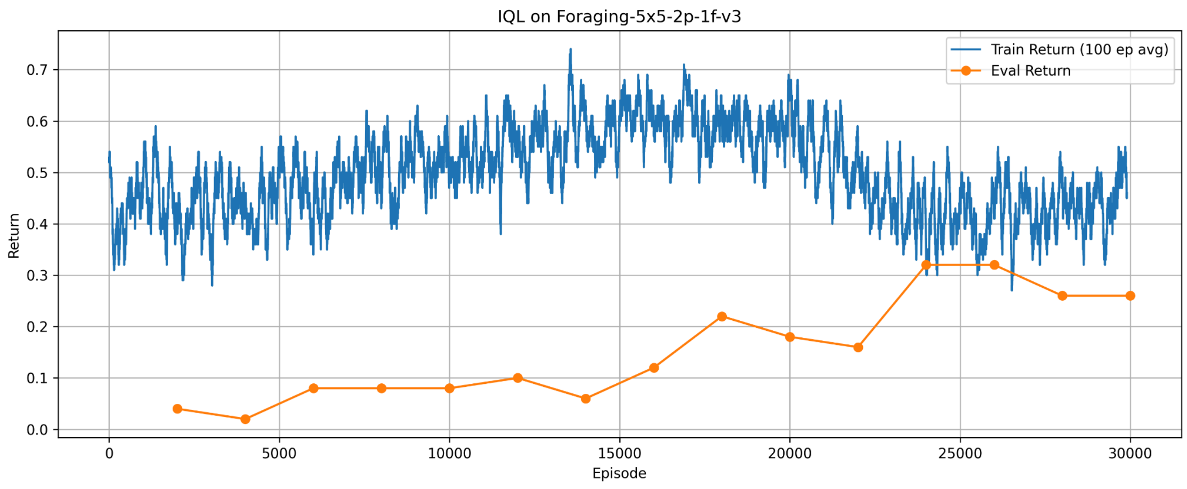
The training returns exhibit extreme variance throughout the entire 30,000 episodes, oscillating between approximately 0.3 and 0.75. This high-frequency noise in the training curve can be unstable learning, likely caused by the non-stationary environment created by both agents simultaneously updating their policies.

On the other hand, the evaluation returns show a more gradual upward trend, improving from near 0.0 to approximately 0.55 by episode 20,000. This could indicate that despite the noisy training process, the agents are slowly learning some coordination strategies. However, the sharp performance collapse at episode 30,000 (dropping to ~0.15) is concerning.



- IQL_Foraging_5x5_2p_1f_v3

The IQL results on the standard environment show similarly high variance in training returns, fluctuating between 0.3 and 0.7 throughout the 30,000 episodes. The evaluation performance improves better than CQL, reaching a peak of approximately 0.32 around episode 25,000. IQL's more conservative.

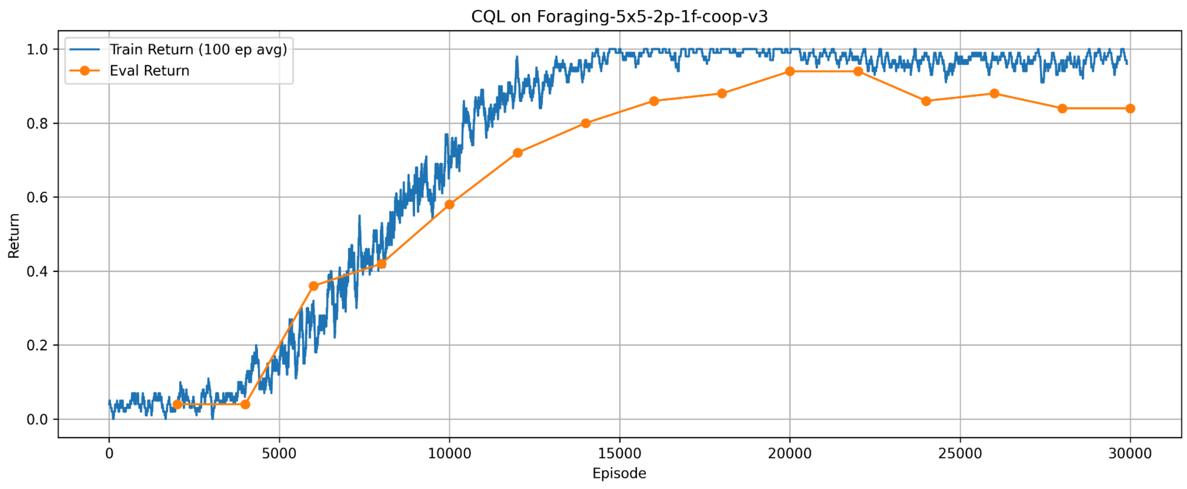


- CQL_Foraging_5x5_2p_1f_coop_v3

The CQL training curves show a smooth and progressive learning process. The training return steadily increases from near 0.0 to approximately 1.0 over the first 15,000 episodes, suggesting that the agents are successfully learning to coordinate and collect food.

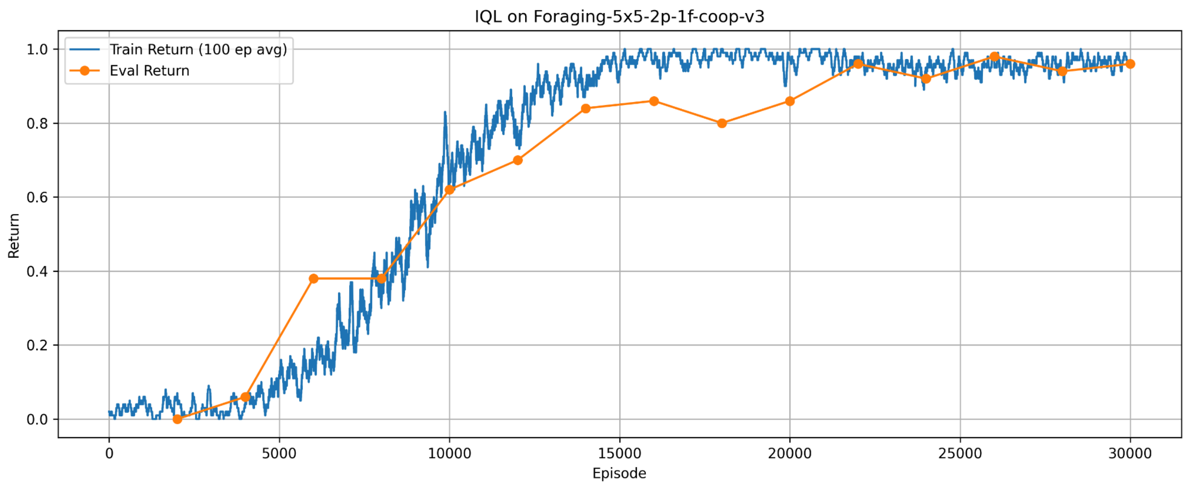
The evaluation return follows a similar upward trajectory, reaching values between 0.85-0.90 and maintaining this performance level through most of the training. This plateau suggests convergence to a near-optimal policy where agents consistently achieve high rewards through cooperation.

The low variance observed in the later training stages indicates that the learned policies are stable and reliable.



- IQL_Foraging_5x5_2p_1f_coop_v3

The plot below shows the IQL results in the same cooperative environment. The learning curves demonstrate remarkably similar behavior to CQL, with training returns increasing from 0.0 to approximately 1.0. IQL appears to converge slightly faster than CQL.
The evaluation returns stabilize between 0.80-0.95, which is comparable to CQL's performance. This suggests that both independent learning approaches are equally capable of solving cooperative tasks when agents' incentives are properly aligned.



In the cooperative environment, both CQL and IQL achieve returns above 0.80, indicating near-optimal coordination. In contrast, the standard environment yields maximum returns of only 0.55 (CQL) and 0.32 (IQL).

3. For each algorithm, provide a detailed explanation and justification of the chosen parameters, describe the training procedure, and discuss the results obtained.

The parameters used were selected to balance exploration with stable convergence in this grid-based environment.

Parameter	**IQL Value**	**CQL Value**
Learning Rate	0.2	0.5
Gamma	0.95	0.95
Total Episodes	30,000	30,000
Epsilon Schedule	0.9 → 0.05	0.9 → 0.05

A learning rate decay was applied during training after 70% of training to fine-tune the learned policy. This was implemented because it was the only thing that ended up working. Initially, I tried many values for the lr, but none of them yielded a good mean reward after x episodes. So, the only thing that worked was to apply this initial lr and then gradually lower it, while doubling the total number of episodes, as the original value was insufficient.

4. Export a video demonstrating a trained agent successfully solving an episode.
Videos are located in the Github repository.
5. For each algorithm, describe the agents' behaviour and relate it to the observed levels of cooperation or competition exhibited by the trained agents.

Standard Environment (Foraging-5x5-2p-1f-v3)

- IQL Agent Behavior

IQL-agents show difficulty reaching the fruit. In some episodes, one of the agents does reach the fruit, but in most cases, they struggle. I had to increase the number of episodes to see the difference compared to CQL, since although CQL also shows difficulty, IQL's somewhat better..

- CQL Agent Behavior

In this case, as mentioned before, this one, like the previous one in the standard environment, has difficulties and only in some episodes are they completed before the step limit set by 50. Looking at the graphs above, these results in this environment may make sense since the CQL proves to be more unstable in the evaluation even though the training is quite similar to the IQL.

Cooperative Environment (Foraging-5x5-2p-1f-coop-v3)

- IQL Agent Behavior

In the cooperative environment, the IQL-trained agents demonstrate highly coordinated foraging behavior. The agents can be observed moving toward food items in a synchronized manner, with both agents positioning themselves adjacent to the same food source before attempting collection. This behavior pattern aligns with the high evaluation returns observed in the training plots, indicating that agents have learned the fundamental requirement of the cooperative variant: food can only be collected when both agents are positioned correctly.

The behavior demonstrates a good level of cooperation, as both agents prioritize joint actions over individual exploration.

- CQL Agent Behavior

The CQL-trained agents exhibit similar cooperative behavior to IQL in this environment. They demonstrate efficient pathfinding toward food items and consistent coordination when collecting resources. Although it is noticeable that the agents are faster and more cooperative than the IQL, they are more effective at ending the episodes.

In summary, CQL agents have shown better results in the **cooperative environment** (although IQL also performs well), and IQL agents have shown better performance in the **standard environment**, although they lose effectiveness in many episodes.

References

- Multi-Agent Reinforcement Learning: Foundations and Modern Approaches. Available at: https://marl-book.com/#download
- *ChatGPT.* used as a support tool for some code development: https://chatgpt.com
- Semitable. *lb-foraging: A Cooperative Multi-Agent Reinforcement Learning Environment*. GitHub repository: https://github.com/semitable/lb-foraging
- *lb-foraging Environment*. GitHub repository: https://github.com/uoe-agents/lb-foraging
- MARL Book Authors. *Multi-Agent Reinforcement Learning Exercises and Code*. GitHub repository: https://github.com/marl-book/marl-book-exercises

- YouTube. *Introduction to Multi-Agent Reinforcement Learning*. Educational video resource: https://www.youtube.com/watch?v=QfYx5q0Q75M