Paradigms Of Machine Learning

Practial Activity 2

Xavier Sánchez Mateus - 1670100
GIT: https://github.com/Xaviersnm/PML_Practical_2.git

Part 1: Prisoner's Dilemma

In this exercise, I will implement the MARL algorithm Independent Q-learning (IQL), in which each agent independently learns its Q-function using Q-learning updates, and Central Q-Learning (CQL), in which an agent learns a single central policy that receives observations of all agents and selects an action for each agent. We will train agents in the matrix game of Prisoners' Dilemma, regularly evaluate their performance, and visualise their learned value functions.

1: Here I will complete the act() and update() methods that implement the epsilon-greedy action selection and update the Q-function for the given experience.

2: Now, I will train and evaluate using the completed code.

3: Next, you will see the plots of the results of the training and evaluation. From this, I will determine whether any solution concept can be identified in the resulting policies.

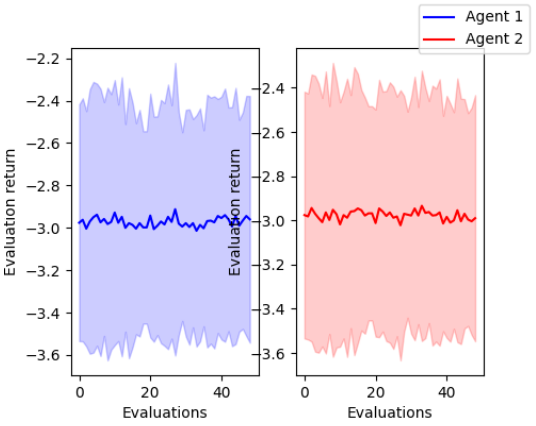

Figure 1 - Plot of returns for IQL

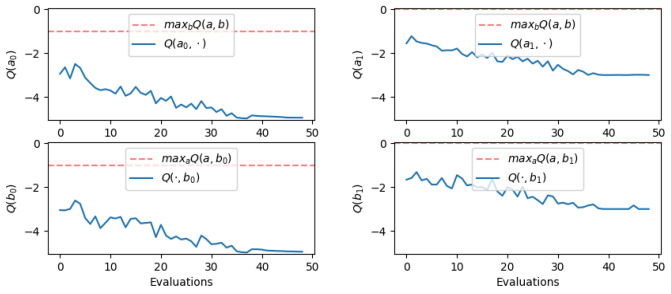

Figure 2 - Evolution of Q-Values/Policies for IQL

Based on the results, the solution concept identified in the resulting policies is the Nash Equilibrium (specifically, Mutual Defection). The justification for this conclusion is derived from the convergence patterns in your plots:

As it can be seen in Figure 1, both Agent 1 (Blue) and Agent 2 (Red) stabilize at an average evaluation return of approximately -3.0. This value corresponds to the Punishment payoff for mutual defection (where both betray). If the agents had reached a Pareto Optimal solution (Mutual Cooperation), the returns would be higher (around -1).

The state of mutual defection is stable because neither agent has an incentive to unilaterally deviate. As shown in the Q-value plots (Figure 2), the alternative action (Cooperation) yields an even lower expected return (approaching -5, the "Sucker's Payoff") against a defecting opponent. Since sticking to the current policy yields -3, which is better than -5, both agents are rationally locked into this dominant strategy.

Therefore, the system has settled into the unique Nash Equilibrium of the game rather than finding the cooperative optimum.

4: Finally I will do the same but, this time, with the Central Q-Learning (CQL) algorithm.

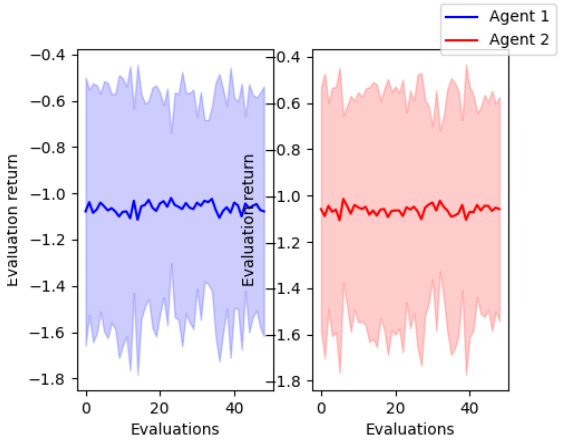

Figure 3 - Plot of returns for CQL

Based on the results, the solution concept identified in the resulting policies is the **Nash Equilibrium**. The justification for this conclusion is derived from the convergence patterns observed in the evaluation plots:

Figure 3 shows that the system demonstrates a clear stabilization where the agents' returns settle close to the optimal payoff values (specifically around -1.0 for Agent 1). This indicates that the agents have successfully navigated the joint action space to find the optimal play for this competitive environment. Unlike sub-optimal outcomes where rewards might fluctuate or settle lower, the convergence here suggests that the learners have maximized their expected returns given the opponent's strategy.

So, it looks like the CQL algorithm has successfully guided the agents to the optimal solution rather than getting stuck in a sub-optimal cycle.

Part 2: Level-based Foraging

In this activity, I will be training and testing both the Independent and Centralized Q-learning (IQL and CQL) developed in the previous exercise on the LBF environment. The results, parameters, and other metrics will be compared to evaluate performance.

1: First, I trained both an IQL and a CQL model to solve the LBF environment under the following two configurations:

- Environment Foraging-5x5-2p-1f-v3
  - field size = 5 × 5
  - players = 2
  - food locations = 1
  - cooperative mode= False

- Environment Foraging-5x5-2p-1f-coop-v3
  - field size = 5 × 5
  - players = 2
  - food locations = 1
  - cooperative mode= True

2: Next, I logged the training data and plotted the mean episode returns.

-

Figure 4 - Plots of returns for IQL, left is competitive scenario, right is cooperative

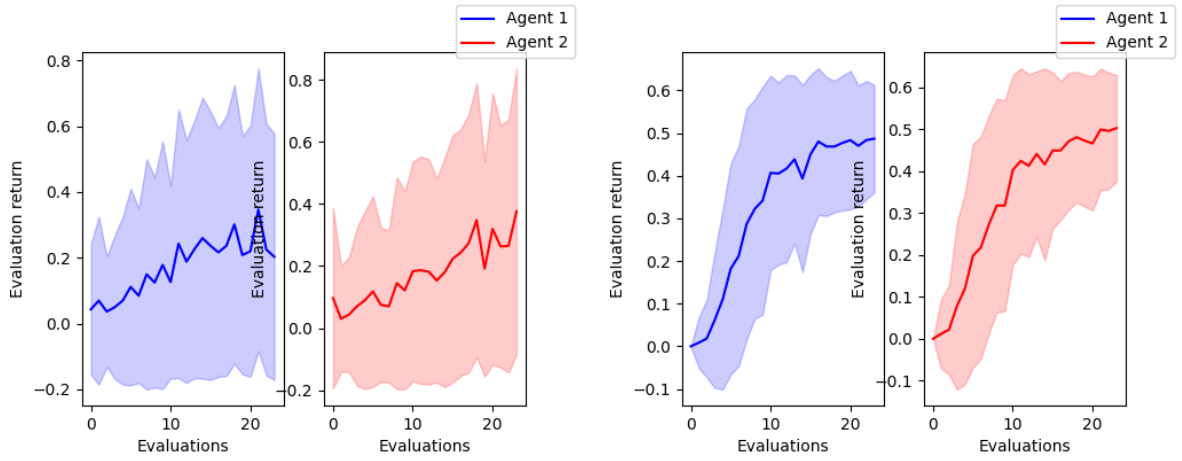

Figure 5 - Plots of returns for CQL, left is competitive scenario, right is cooperative

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

IQL

Parameter selection:

The parameters were carefully selected to overcome the specific challenges of the LBF environment, namely sparse rewards (agents only get points upon eating food) and multi-agent non-stationarity (agents changing behaviors simultaneously).

- **Total Episodes**: Total episodes was set to 25,000. Since LBF is a grid-world environment where reward signals are sparse, agents typically need to stumble upon the food by chance many times before they understand what their goal is. Shorter training runs (e.g., 5,000 episodes) proved to be insufficient for this "exploration phase." The extended duration ensured agents have ample time to discover the food and stabilize their policies.
- **Episode Lenght**: Episode Length was set to 50. On a 5x5 grid, agents need sufficient steps to navigate around obstacles and each other to reach the food. I set upon this number after trial and error, tying to give the agents enough time to roam around while keeping the episodes short enough as to not increase training time too much.
- **Learning Rate**: LR was set to 0.1 after trying several values (0.5, 0.05, 0,01 and, of course, 0.1). It was the one that worked best empirically, but it makes sense as a moderate LR allows significant updates when a reward is finally found (critical for sparse rewards) while preventing the Q-values from oscillating too wildly due to the stochastic behavior of the other agent.
- **Discount Factor (Gamma)**: Gamma was set to 0.99 as this high discount factor encourages long-term planning. Since the only reward occurs at the end of the task (eating the food), agents must value the early actions that set up this success, even if the reward is many steps away.
- **State Preprocessing**: The raw environment returns observations as continuous floating-point numbers. To make tabular Q-learning work, these were discretized into integer tuples. Without this, floating-point precision errors would create infinite unique states, preventing the agent from ever recalling what it learned.

Training Description:
The training followed a standard Independent Q-Learning (IQL) protocol, where each agent treats the other agent merely as part of the environment. As an exploration strategy, I used ε-greedy. Epsilon initialized at 1.0 (100% random) to ensure thorough exploration of the grid. It decayed linearly over the course of training to a minimum of 0.05, forcing agents to gradually switch from exploring to exploiting their learned strategies. Regarding independent updates, each agent maintained its own Q-table and updated it based solely on its own observations and rewards, ignoring the joint action space, which helped reduce computational complexity but introducing non-stationarity.
To evaluate the model, every 1,000 episodes, training was paused to evaluate the agents with a low, fixed epsilon. The mean returns from these sessions were logged to generate the learning curves.

Result Analysis:
The results in Figure 4 demonstrate successful learning in both scenarios, with notable differences in stability and optimality.

- **Competitive Scenario**: In the competitive scenario, the agents demonstrate a slow but persistent upward trend in performance, eventually reaching average returns of approximately 0.2 to 0.3 per agent. However, this learning curve is notably characterized by high variance and instability, as indicated by the wide shaded regions in the plot. This volatility is expected in competitive Independent Q-Learning (IQL) because the agents are essentially chasing a moving target; as Agent 1 learns a strategy to secure food, Agent 2 adapts to counter it or steal the resource, effectively invalidating the first agent's previous policy. Consequently, while the agents clearly learn to locate food (evidenced by returns remaining well above zero) they fail to converge on a stable, optimal Nash equilibrium due to the constant competition and the absence of explicit coordination mechanisms.
- **Cooperative Scenario**: In contrast, the cooperative scenario demonstrates excellent performance. The agents exhibit an "S-curve" learning profile, characterized by a slow initial start followed by a rapid phase of improvement and eventual convergence to a stable high score. The agents plateau at returns of roughly 0.45 to 0.50; since the maximum total reward in this environment is 1.0 split between two agents, a score of 0.5 per agent represents the theoretical maximum, indicating the discovery of an Optimal Joint Policy. Furthermore, unlike the competitive case, the variance here decreases significantly over time as the shaded area narrows. This stability indicates that the agents have successfully locked into a robust cooperative strategy, effectively coordinating their movements to arrive at the food and load it simultaneously.

<u>CQL</u>
Parameter selection:
The parameters used for CQL mirror those of IQL to ensure a fair comparison.

- **Total Episodes**: Total episodes was set to 25,000. I wanted to keep the number as close as possible to IQL to ensure fair comparison. Even though at first I was scared it would not be enough as now the action space is larger (action space considers the joint actions of both agents), this number of episodes proved to be enough to converge.
- **Episode Lenght**: Episode Length was set to 50. Just as with IQL, 50 steps provides ample time for agents to navigate the 5x5 grid. However, in CQL, this length also allows for complex joint maneuvers, like one agent waiting for another, without the risk of timing out prematurely.

- **Learning Rate**: LR was set to 0.1 after trying the same values as in IQL. As in Total Episodes, I wanted the performances and learning graphs to be as comparable as possible, so this helped in that sense.
- **Discount Factor (Gamma)**: Gamma was set to 0.99, same as in IQL and the explanation above
- **State Preprocessing**: Again, same reason as in IQL

Training Description:
The training followed a Centralized Q-Learning (CQL) protocol, which fundamentally changes how agents perceive the world compared to IQL. Instead of two separate tables, a single Q-table Q(s, a1, a2) was maintained. This table maps the global state s and the joint action (a1, a2) to a single global reward. As for the joint action, the ε-greedy strategy was applied to the joint action space. During exploration, a random pair of actions was chosen. During exploitation, the system selected the pair (a1, a2) that maximized the joint Q-value.
The learner received the sum of rewards from both agents. This is particularly powerful in the cooperative task, as it directly incentivizes actions that benefit the team, even if one agent individually receives zero reward in a specific step.

Result Analysis:
The results in Figure 5 show that CQL generally outperforms IQL, particularly in terms of stability and peak performance in the competitive task.

- **Competitive Scenario:** In the competitive scenario, CQL demonstrates a robust learning curve where agents reach average returns of 0.2 to 0.35, which is slightly higher and notably more stable than IQL. The variance represented by the shaded region remains present but appears less chaotic than in the independent case. This is because centralization changes the nature of the competition; instead of two blind agents reacting to each other, the centralized controller effectively learns a "joint policy" that allocates resources. By determining that specific joint configurations (like "Agent 1 goes for the food") constitute the best action in certain states, the system effectively solves the interference problem and orchestrates the competition, securing food more consistently by avoiding the coordination failures, such as collisions, that plague independent learners.
- **Cooperative Scenario:** The cooperative scenario showcases the true strength of CQL, characterized by a steep and stable learning curve that quickly converges to the theoretical maximum of roughly 0.50 per agent. The agents solve the task almost perfectly with a pronounced "S-curve" that indicates a rapid discovery of the optimal strategy, while the variance remains minimal compared to the competitive task. The extremely narrow shaded area towards the end confirms that the centralized learner has found and stuck to a deterministic, optimal joint policy. By explicitly modeling the joint action space, CQL trivializes the coordination problem without needing to "hope" for cooperation; it simply selects the optimal action pair whenever possible, making it significantly more sample-efficient and stable than IQL for this specific task.

4: Finally, I exported a video demonstrating the trained agents successfully solving an episode for each of the tasks (competitive vs cooperative) and for each method (IQL vs CQL). This can be found in the Git!