6FTC2086 Practical Assignment

Semester A 2024/2025

There are 35 marks available, with each mark representing 1% of your total grade for the module.

Submission requirements

You may discuss your general ideas and thoughts with peers but the work handed in must be distinctly yours and your own. The following sets of documents must be submitted through canvas as individual files, not a directory. (If you have many Python files please merge this down into 3 or less.)

(a) Your code that runs the different simulations
(b) The csv-files for each simulation
(c) A report to support the simulations

Objective

Implement a simple reinforcement learning (RL) algorithm (Q-learning) in a Multi-Agent System (MAS) to allow agents to learn optimal behaviors in a grid environment.

Task 1: Implement Q-Learning for Agents (15 Marks)

1. **Q-Table and Agent Class (9 Marks)**
    - Implement an Agent class that interacts with the environment by learning from the rewards received. Each agent should maintain a Q-table, which maps states to actions. (2 Marks)
    - Define the following key methods:
        - Q-Table Initialization: Initialize Q-values for state-action pairs. (2 marks)
        - Action Selection: Use the epsilon-greedy method to choose an action (explore or exploit). (2 Marks)
        - Q-Update Rule: Implement the Q-learning update rule to adjust Q-values based on rewards. (2 Marks)
        - Rewards: Define rewards in the grid as follows: +20 for reaching the goal, and -1 for each move made that does not reach the goal. (1 Marks)
2. **Environment Setup (6 Marks)**
    - Create a 10x10 grid environment where agents can move up, down, left, or right. (2 Marks)
    - Define the boundary conditions for the grid by ensuring that the agent cannot move outside the grid’s walls. Additionally, place four obstacles within the

grid that the agent cannot pass through, creating barriers that the agent must navigate around while learning to reach its goal. (2 Marks)

- Set the target goal as the agent reaching the bottom-right corner of the grid. (2 Marks)

Task 2: Run the Simulation (10 Marks)

1. **Simulate Learning (5 Marks)**
    - Run the simulation for 200, 800, and 1000 episodes with a single agent learning to navigate the grid.
    - Track the agent's performance by storing the episode length and the total reward accumulated in each episode.
2. **Multi-Agent Simulation (5 Marks)**
    - Expand the simulation to include 4 agents learning concurrently in the grid.(2 Marks)
    - Use shared or independent Q-tables for agents and analyze their behavior (whether they interfere with or help each other). (2 Marks)
    - Store the position and Q-values for each agent at the end of every episode. (1 Marks)

Task 3: Analyze the Results and Report (10 Marks)

- Compare how quickly the agent(s) learn to reach the goal by analyzing the total rewards and episode lengths over time. (2 Marks)
- Plot graphs showing the learning curves (total reward per episode) for both single-agent and multi-agent scenarios. (2 Marks)
- Discuss the behavior of multiple agents learning together: do they compete for the goal or converge on different strategies and paths? (2 Marks)
- Write a report detailing the implementation, simulation analysis and results. (4 Marks)

Submission Requirements:

Ensure the following items are included in your submission:

1. Python Code

- **Single-Agent Simulation Code:**
    - Q-learning implementation for a single agent.
    - Includes Q-table initialization, action selection (epsilon-greedy), and Q-value updates.
    - Agent's movement in a grid environment with rewards and penalties.

- **Multi-Agent Simulation Code:**
    - Code expanded to include 3 agents.

- Each agent learns concurrently in the grid with either shared or independent Q-tables.
- Agents interact with the environment and learn optimal behaviors.

2. CSV Files

- **Single-Agent Simulation:**
  - Store the following data for each episode:
    - Episode number.
    - Total reward accumulated.
    - Episode length (number of steps).
- **Multi-Agent Simulation:**
  - Store the following data for each agent at the end of each episode:
    - Agent positions.
    - Q-values for the episode.
    - Total rewards accumulated for each agent.

3. Report (PDF Format)

- **Introduction:**
  - Briefly explain the objectives of the project and an overview of Q-learning and reinforcement learning.
- **Q-Learning Parameters:**
  - Explain the selected parameters (learning rate, discount factor, exploration rate) and their impact on learning.
- **Single-Agent Learning Behavior:**
  - Discuss the agent's performance over time (e.g., total rewards, episode length).
  - Include a graph showing the learning curve (total reward per episode).
- **Multi-Agent Learning Behavior:**
  - Analyze the learning performance of multiple agents.
  - Discuss any observed collaboration or competition between agents.
  - Include a graph showing the total rewards over episodes for each agent.
- **Conclusion:**
  - Summarize the results and discuss the efficiency of the learning process.
  - Mention potential improvements or future directions.