Laboratory 6

Variant 2

Group 12

By Dimitrios Georgousis and Andrea Amato

Introduction

This mini-project aims to familiarize students with the concept of Q-Learning within reinforcement learning. Specifically, we were tasked with implementing a Q-Learning algorithm to solve the "MountainCar-v0" problem from the Gymnasium library. In this environment, the agent (a car) must drive up a steep hill to reach a specified goal position. However, the car's engine is underpowered, and thus the agent needs to build momentum by moving back and forth. The main challenges include:

- **Continuous State Space**: The position and velocity of the car are continuous variables.
- **Sparse Reward Structure**: The agent only receives a reward of -1 for each time step until reaching the goal.

Model Description

We employed a discretized version of the state space to transform continuous states into a manageable form for our Q-table. The agent was trained using the following hyperparameters:

- **Learning Rate (α)**:
    - Controls the step size in the Q-value update.
    - Set to values like 0.1, 0.3, and 0.5 for experimentation.
- **Discount Factor (γ)**:
    - Balances immediate and future rewards.
    - Chosen values were 0.7 and 0.9.
- **Epsilon (ε)**:
    - Governs the trade-off between exploration and exploitation.
    - Set to 1 initially and decayed exponentially over time to a minimum of 0.01.

The Q-table was initialized with random values to encourage exploratory actions in the initial episodes. The agent interacts with the environment as follows:

1. **State Discretization**:
    - The continuous state space of MountainCar-v0, consisting of position and velocity, is discretized into bins to allow indexing in the Q-table.
    - The position is divided into `N_POSITIONS` bins, while the velocity is divided into `N_VELOCITIES` bins.
2. **Action Space**:
    - The environment has three discrete actions: 0 (push left), 1 (no push), and 2 (push right).
3. **Reward Structure**:
    - The agent receives a reward of -1 at each time step until it reaches the goal at $x_{car} \geq 0.5$.

Training Procedure

In the "MountainCar-v0" environment, the agent interacts with the environment over thousands of episodes, with each episode consisting of multiple steps where actions are chosen, rewards are received, and Q-values are updated. Here's a step-by-step breakdown of how the training procedure works:

1. **Initialization**:
    - **Q-Table Initialization**:
        - The Q-table is initialized with random values in the range [-2, 0] for all state-action pairs, therefore encouraging initial exploration.
    - **Parameter Initialization**:
        - Key parameters like learning rate (α), discount factor (γ), epsilon (ε), and epsilon decay are set based on the experimentation configuration.
2. **Episode Loop**:
    - The training loop iterates over the specified number of episodes (`N_EPISODES`).
3. **Inside Each Episode**:
    - **Reset Environment**:
        - The environment is reset to an initial state.
        - The initial state is discretized using predefined bins for both position and velocity.

- **Initialize Variables**:
    - A `done` flag signaling whether the agent has reached the termination state is set to **False**.
    - Training metrics such as `total_reward` and `episode_length` are initialized to 0.
- **Step Loop**:
    - Each episode runs up to a maximum number of steps or until the agent reaches the goal.
    - **Action Selection (Epsilon-Greedy Strategy)**:
        - With probability ε, the agent explores by choosing a random action.
        - Otherwise, it exploits by selecting the action with the highest Q-value for the current state.
    - **Execute Action**:
        - The chosen action is executed in the environment using the `render_env.step(action)` function.
        - This returns the next state, reward, and a boolean indicating if the episode has ended.
        - The next state is also discretized.
    - **Q-Value Update**:
        - The Q-value for the current state-action pair is updated using the Bellman equation:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t))$$

          Where:
            - s_t and s_{t+1} are the current and next states.
            - a_t is the action taken at state s_t.
            - $r_{t+1}$is the reward received after taking action a_t.
            - α is the learning rate
            - γ is the discount factor
            - $\max_a Q(s_{t+1}, a_t)$ is the estimated optimal future value.
        - If the terminal state is reached, the reward is set to zero.

- **Update Variables**:
    - We update the collected metrics `total_reward` and `episode_length`
    - Set the current state to the next state.

- **End of Episode**:
    - **Epsilon decay**: The exploration rate is decayed exponentially to promote exploitation in later episodes:

$$\varepsilon \leftarrow \max\left(\varepsilon_{min}, \exp\left(-\varepsilon_{decay} \cdot episode\right)\right)$$

Results

The results are organized into three primary categories: total rewards per episode, episode lengths, and convergence rates. These metrics were logged and visualized to understand the agent's learning progression and to fine-tune the hyperparameters.

1. Total Rewards

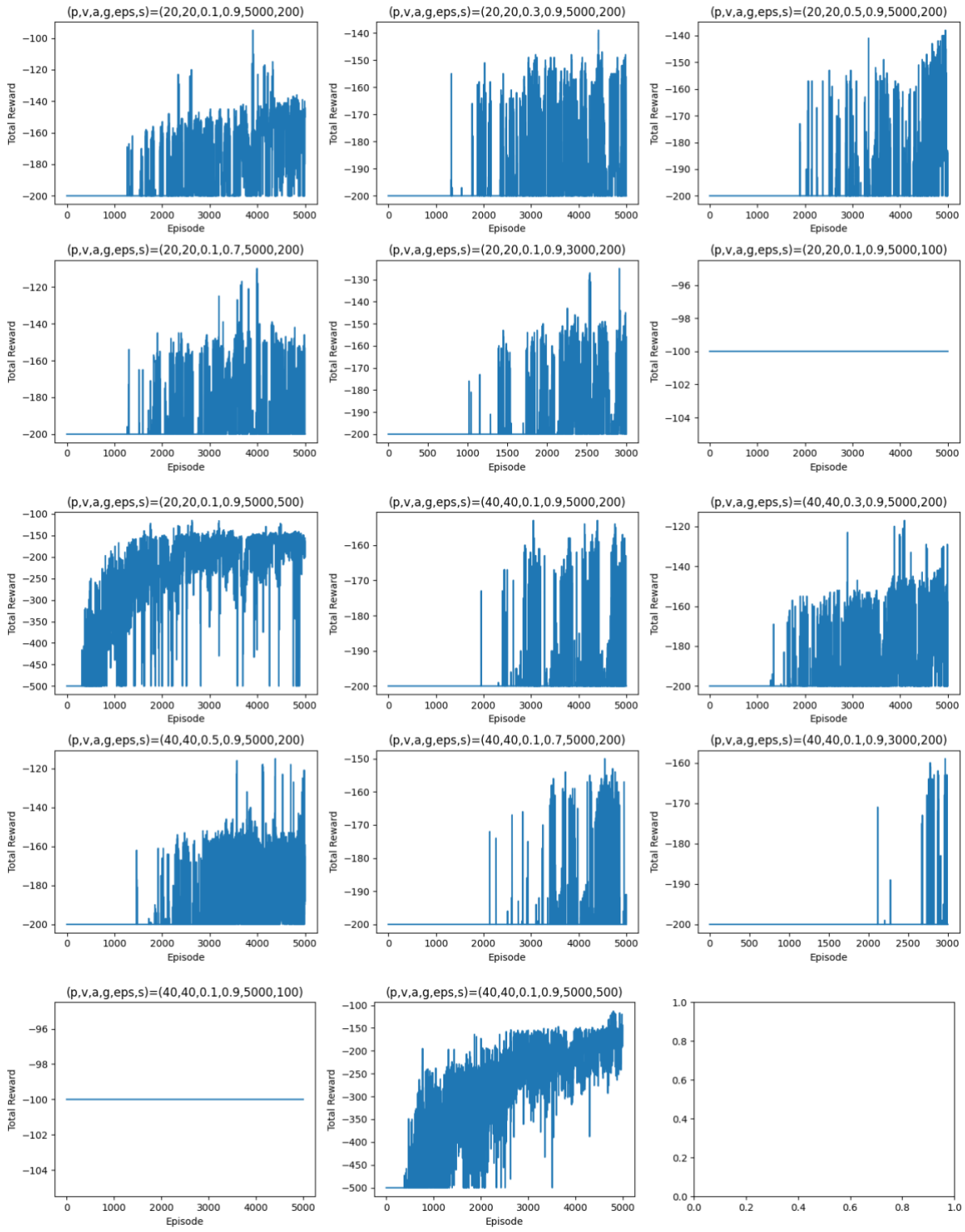

Figure 1 – Total rewards per episode (varying configurations)

Description: These graphs show the total rewards accumulated by the agent in each episode. Each subplot shows the results for a different configuration. We tested combinations of different hyperparameters like learning rate (α), discount factor (γ), and the number of episodes.

Comments on the plots:

- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 100):** The rewards remain constant at -100, indicating that the agent was unable to solve the problem due to the limited number of steps compared to the default configuration.
- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 200):** A steady increase in rewards is visible, with rewards reaching around -150, demonstrating the agent's learning progress. Comparable to the default configuration.
- **(p, v, a, g, eps, s) = (40, 40, 0.3, 0.9, 5000, 200):** Rewards fluctuate heavily in the initial episodes but stabilize after episode 2000, indicating the agent's gradual improvement over time. More volatile compared to the default configuration due to the higher learning rate.
- **(p, v, a, g, eps, s) = (40, 40, 0.5, 0.9, 5000, 200):** The agent shows rapid improvement and maintains rewards above -200 consistently. The higher learning rate helps convergence but introduces increased volatility compared to the default configuration.
- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.7, 5000, 200):** The agent struggles initially but demonstrates consistent progress, stabilizing around -140. The lower discount factor reduces long-term planning capabilities compared to the default configuration.
- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 5000, 500):** The agent starts with very low total rewards, showing difficulty in solving the task initially. Over the episodes, the agent gradually improves, achieving an average reward of around -100 by the end. Comparable to the default configuration.
- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 3000, 200):** Initially, the agent struggles with low rewards but improves gradually over the episodes. The agent eventually stabilizes at around -160, indicating reasonable learning despite the lower number of episodes.
- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 5000, 200)**: The agent shows a steady increase in total rewards, stabilizing at around -140 by the end of the episodes. Similar to the default configuration.
- **(p, v, a, g, eps, s) = (20, 20, 0.3, 0.9, 3000, 200)**: The agent demonstrates high volatility initially due to the high learning rate, before stabilizing around -160. The reduced number of episodes impacts the convergence rate compared to the default configuration.
- **(p, v, a, g, eps, s) = (20, 20, 0.5, 0.9, 3000, 200)**: The agent initially struggles but rapidly improves, stabilizing at around -140. The high learning rate allows for faster convergence but also introduces increased volatility compared to the default configuration.
- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.7, 5000, 200)**: The agent stabilizes at around -140 after fluctuating initially. The lower discount factor reduces the agent's ability to plan for future rewards, resulting in longer episode lengths compared to the default configuration.

- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 3000, 200)**: The agent shows gradual improvements in total rewards, stabilizing at around -160. Despite fewer episodes, the agent manages to converge steadily compared to the default configuration.
- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 500)**: The agent begins with low rewards but gradually improves, achieving an average reward of around -100. The high step limit and learning rate provide sufficient flexibility for exploration compared to the default configuration.
- **(p, v, a, g, eps, s) = (40, 40, 0.3, 0.9, 5000, 200)**: Initially, the agent struggles but gradually reduces episode lengths, stabilizing around -120. The higher learning rate helps the agent converge faster but introduces initial volatility compared to the default configuration.

2. Episode Lengths

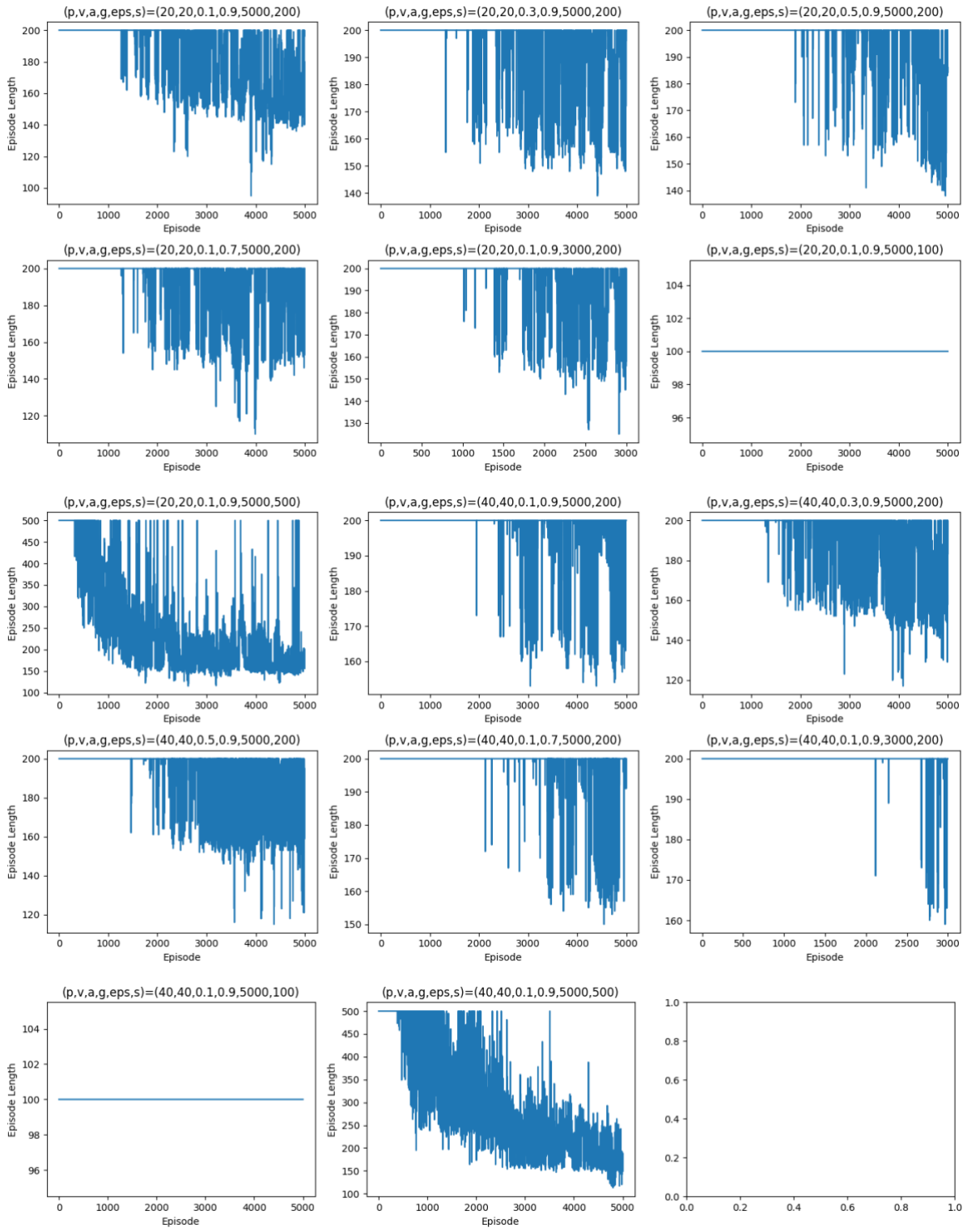

Figure 2 – Episode lengths per episode (varying configurations)

Description: These graphs show the number of steps the agent took in each episode. A decreasing trend in episode length indicates that the agent is learning to solve the task more quickly.

Comments on the plots:

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 100):** Episode length remains constant at 100, indicating that the agent was unable to solve the problem due to the limited number of steps compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 200):** The agent initially struggles but gradually reduces episode lengths, stabilizing around 150. Comparable to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.3, 0.9, 5000, 200):** Episode lengths fluctuate heavily in the initial episodes, indicating that the agent is struggling to learn. However, the agent stabilizes around episode 2000, showing gradual improvement over time. More volatile compared to the default configuration due to the higher learning rate.

• **(p, v, a, g, eps, s) = (40, 40, 0.5, 0.9, 5000, 200):** The agent shows rapid improvement, maintaining episode lengths below 150 consistently. The higher learning rate helps with convergence but introduces increased volatility compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.7, 5000, 200):** The agent demonstrates consistent progress, stabilizing around 140, but takes longer to reach convergence due to the lower discount factor. The longer episode lengths compared to the default configuration result from reduced long-term planning capabilities.

• **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 5000, 500):** The agent initially struggles with high episode lengths, but the learning rate of 0.1 allows for steady learning. The agent gradually stabilizes around episode 3000, reducing episode lengths to approximately 150. Comparable to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 3000, 200):** Initially, the agent struggles with longer episode lengths but improves gradually, stabilizing at around 150. Despite fewer episodes, the agent shows reasonable learning compared to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 5000, 200):** The agent shows a steady reduction in episode lengths, stabilizing at around 140 by the end of the episodes. Similar to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.3, 0.9, 3000, 200):** The agent demonstrates high volatility initially due to the high learning rate but eventually stabilizes around 160. The reduced number of episodes impacts the convergence rate compared to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.5, 0.9, 3000, 200):** The agent initially struggles but rapidly improves, stabilizing around 140. The higher learning rate allows for faster convergence but also introduces increased volatility compared to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.7, 5000, 200):** The agent stabilizes at around 140 after fluctuating initially. The lower discount factor reduces the agent's ability to plan for future rewards, resulting in longer episode lengths compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 3000, 200):** The agent shows gradual reductions in episode lengths, stabilizing around 160. Despite fewer episodes, the agent manages to converge steadily compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 500):** The agent begins with high episode lengths but gradually improves, achieving an average episode length of around 100. The high step limit and learning rate provide sufficient flexibility for exploration compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.3, 0.9, 5000, 200):** Initially, the agent struggles but gradually reduces episode lengths, stabilizing around 120. The higher learning rate helps the agent converge faster but introduces initial volatility compared to the default configuration.

3. Convergence Rates

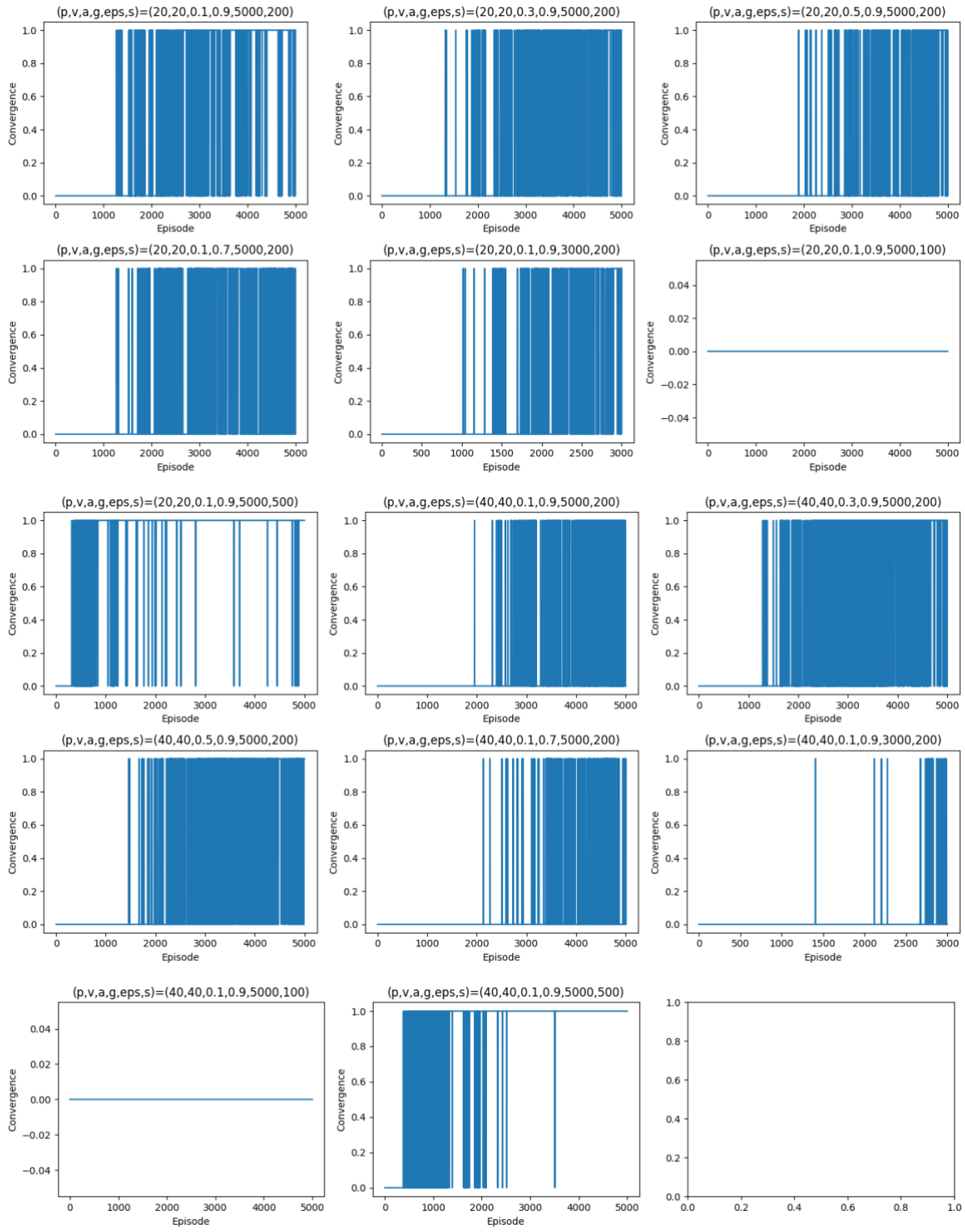

Figure 3 – Convergence rates per episode (varying configurations)

Description: These graphs display the episodes in which the agent reached the goal. Peaks in this graph represent successful episodes where the agent reached the goal within the maximum allowed steps.

Comments on the plots:

- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 100):**

The convergence rate remains at zero throughout the episodes, indicating that the agent failed to solve the problem due to the limited number of steps compared to the default configuration.

- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 500):**

The agent achieves convergence several times across episodes, with early fluctuations gradually stabilizing after episode 1500. The higher step count (500) allows the agent to explore more efficiently compared to the default configuration.

- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 200):**

The agent converges consistently after episode 1000, indicating steady learning progress. Convergence stabilizes toward the later episodes compared to the default configuration.

- **(p, v, a, g, eps, s) = (40, 40, 0.3, 0.9, 5000, 200):**

The agent initially exhibits high fluctuation in convergence rates due to the higher learning rate but stabilizes around episode 2000. The higher learning rate allows for faster convergence than the default configuration but with increased volatility.

- **(p, v, a, g, eps, s) = (40, 40, 0.5, 0.9, 5000, 200):**

The agent converges consistently after episode 1000, though with higher volatility in early episodes. The high learning rate (0.5) promotes faster convergence compared to the default configuration.

- **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.7, 5000, 200):**

Convergence is achieved gradually after fluctuating in the initial episodes. The lower discount factor (gamma = 0.7) causes longer convergence time compared to the default configuration.

- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 5000, 500):**

The agent converges after episode 1500 with minimal fluctuations in later episodes. The higher step limit allows for efficient exploration, similar to the default configuration.

- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 3000, 200):**

The agent initially exhibits low convergence but improves steadily over episodes. The convergence rate stabilizes toward the end of the episodes, comparable to the default configuration.

- **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.9, 5000, 200):**

The agent stabilizes convergence after episode 1000, showing consistent learning progress. The results are similar to the default configuration.

- **(p, v, a, g, eps, s) = (20, 20, 0.3, 0.9, 3000, 200):**

The agent exhibits high fluctuation due to the high learning rate but stabilizes after episode 1500. The reduced number of episodes affects convergence rates compared to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.5, 0.9, 3000, 200):**
The agent initially shows high fluctuation due to the high learning rate but achieves convergence gradually. The reduced number of episodes affects convergence compared to the default configuration.

• **(p, v, a, g, eps, s) = (20, 20, 0.1, 0.7, 5000, 200):**
The agent achieves convergence gradually after fluctuating initially. The lower discount factor (gamma = 0.7) reduces planning for future rewards, affecting convergence compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 3000, 200):**
The agent shows gradual improvement in convergence rates after initial fluctuations. Despite fewer episodes (3000), the agent manages steady convergence compared to the default configuration.

• **(p, v, a, g, eps, s) = (40, 40, 0.1, 0.9, 5000, 500):**
The agent begins with low convergence rates but gradually improves, achieving consistent convergence after episode 1000. The higher step limit and learning rate provide sufficient flexibility for exploration compared to the default configuration.

Discussion on results

The agent showed significant improvement in its ability to reach the goal as evidenced by the increasing total rewards and decreasing episode lengths. The epsilon decay played a critical role in balancing exploration with exploitation, allowing the agent to discover efficient paths to the goal.

Parameter tuning revealed that a moderate learning rate and discount factor facilitated the best performance, providing a good balance between learning quickly and stabilizing the updates to the Q-table. The exploration rate (epsilon) needed careful adjustment to ensure sufficient exploration during the initial phase and increased exploitation as the agent's policy matured.

Parameter Tuning Observations:

- **Learning Rate (α)**:
    - Moderate learning rates (e.g., 0.1) provided stable learning.
    - High learning rates (e.g., 0.5) led to faster convergence but could also cause instability.
- **Discount Factor (γ)**:
    - A high discount factor (0.9) promoted long-term reward optimization but required more episodes for convergence.

    - A moderate discount factor (0.7) allowed for quicker convergence but at the expense of overall rewards.

- **Epsilon (ε)**:
    - A well-decayed epsilon ensured efficient exploration initially and focused exploitation later.

Conclusions

The Q-Learning algorithm proved effective for the "MountainCar-v0" problem, with the agent learning to reach the goal consistently after several episodes. This exercise has highlighted the importance of parameter tuning in reinforcement learning and demonstrated the utility of Q-Learning in environments with sparse rewards and continuous state spaces. Future work could experiment with different reward structures, such as providing additional rewards for reaching intermediary points, which could possibly accelerate convergence.