5x5 Grid Navigation Problem

By Karingattil Sagar Thomas , Vitthal Vinod Tiwary

2022A7PS0156H , 2022A3PS0553H

Problem Statement

We have chosen a problem involving navigating a 5x5 grid, starting from the top-left corner at position (0,0) with the objective of reaching the bottom-right corner at position (4,4). The grid contains special cells with distinct properties:

- **Portals:** Located at positions (0,3), (3,3), and (1,0), these cells transport you randomly to one of the other two portal locations or remain on the same location with equal probability upon entering. Each time a portal is used, a reward of -3 is incurred.
- **Kill State:** The cell at position (3,2) is a "kill state." Moving to this cell results in a penalty of -10 and teleports you back to the starting position (0,0).
- **Regular Cells:** Moving to any other cell (except the goal) results in a reward of -1.

Grid Layout

R\C	0	1	2	3	4
0	S			O	
1	O				
2					
3			X	O	
4					T

Table 1: 5x5 Grid Layout

On-Policy Monte Carlo

In an on-policy Monte Carlo method, we aim to improve the policy we are following. This method evaluates and improves the same policy that is used to generate the episodes.

The target policy for my problem is **epsilon-greedy**, which balances exploration and exploitation. We have chosen $\epsilon = 0.2$, meaning the agent will exploit the best-known action 80% of the time and explore other actions 20% of the time.

The epsilon-greedy policy is defined as follows:

$$\pi(s,a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a = \arg\max_{a'} Q(s,a') \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{otherwise} \end{cases}$$

where:

- $\pi(s,a)$ is the probability of taking action a in state s under the policy.
- $Q(s,a')$ is the action-value function, representing the expected return for taking action a' in state s.
- $\epsilon = 0.2$ is the exploration parameter, controlling the balance between exploration and exploitation.
- $|\mathcal{A}(s)|$ is the number of actions available in state s.

Additionally, the discounting factor γ has been set to 0.8. This discount factor reduces the weight of future rewards, prioritizing nearer-term rewards in the calculation of expected returns:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

In this case, $\gamma = 0.8$ determines how much future rewards are discounted over time.

Off-Policy Monte Carlo with Weighted Importance Sampling

In an off-policy Monte Carlo method, we evaluate and improve a target policy π using data generated by a different behavior policy b. This method allows for the improvement of a policy that is not the one generating the experience.

For this problem, We will use **weighted importance sampling** to estimate the value of the target policy π from the behavior policy b. The expected return G is weighted by the ratio of the probabilities of the trajectories under the target and behavior policies.

The importance sampling ratio for an episode is given by:

$$\rho_T = \prod_{t=0}^{T-1} \frac{\pi(a_t|s_t)}{b(a_t|s_t)}$$

where:

- $\pi(a_t|s_t)$ is the probability of taking action a_t in state s_t under the target policy π.
- $b(a_t|s_t)$ is the probability of taking action a_t in state s_t under the behavior policy b.

Using **weighted importance sampling**, the value of the target policy is estimated by adjusting the returns using the importance sampling ratio:

$$\hat{v}(s) = \frac{\sum_{k=1}^{n} \rho_T^{(k)} G^{(k)}}{\sum_{k=1}^{n} \rho_T^{(k)}}$$

where:

- $\hat{v}(s)$ is the estimated value of state s under the target policy π.
- $G^{(k)}$ is the return (total discounted reward) following the k-th episode.
- $\rho_T^{(k)}$ is the importance sampling ratio for the k-th episode.

This method corrects for the differences between the behavior and target policies, ensuring that the target policy π is accurately evaluated despite being off-policy.

1 Summary of Code Functionality

1.1 Initialization

The grid is defined with various states including a goal state, portal states, and a kill state. Rewards are set for reaching the goal, portal states, and the kill state, as well as the discount factor γ and exploration parameter ϵ.

1.2 Functions

- `get_next_state(state, action)`: Determines the next state based on the current state and action.
- `generate_episode(policy)`: Generates an episode following the given policy. Handles state transitions, rewards, and special states (kill and portal).
- `find_first_occurrence_index(episode, target_tuple)`: Finds the first occurrence of a state-action pair in an episode.
- `sum_rewards_from_index(episode, start_index)`: Computes the total discounted reward from a given start index in an episode.

1.3 On-Policy Method

- Initializes the state-value function V and a policy.
- Updates the policy to be greedy with respect to the value function.

- Generates episodes and updates the state-value function using returns from each state-action pair.
- Returns the optimal state-value function V and the final policy.

1.4 Off-Policy Method

- Initializes the action-value function Q, behavior policy, and target policy.
- Updates the target policy to be greedy with respect to the action-value function.
- Behavioral Policy is random always with 0.25b probabilty in each direction
- Generates episodes and updates the action-value function using the importance sampling ratio ρ.
- Returns the optimal action-value function Q and the final target policy.

1.5 Policy Check

- Verifies the optimality of the policies obtained from on-policy and off-policy methods by generating episodes and computing average rewards and episode lengths.

1.6 Data Collection and Visualization

- Runs experiments with varying numbers of episodes to collect data on the performance of both policies.
- Plots graphs comparing on-policy and off-policy results for value functions, rewards, and average episode lengths.
- Fits and plots best-fit lines and curves to analyze the relationship between the number of episodes and performance metrics.

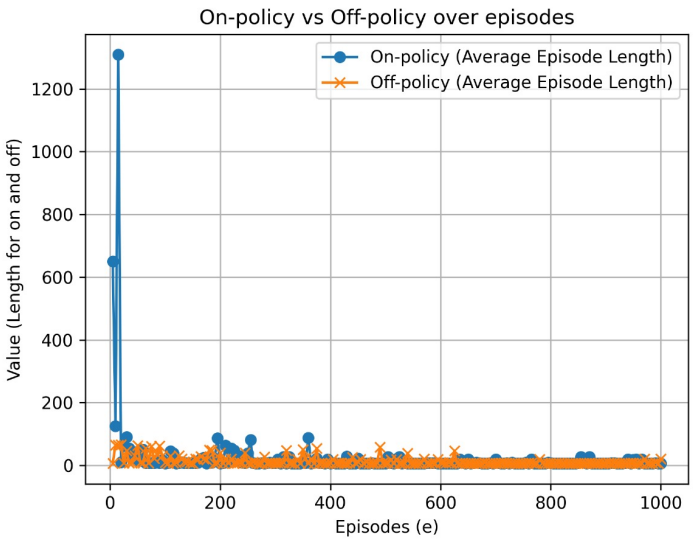

Figure 1: On policy and Off Policy Length of each epiosde

Analysis of On-Policy vs Off-Policy Approaches

Graph 1: On-Policy vs Off-Policy (Average Episode Length)

Observation: On-policy starts with very high episode lengths (spikes above 1200) in the initial episodes and then stabilizes. Off-policy, on the other hand, maintains a more stable and consistently lower episode length from the beginning.

Analysis: This suggests that the on-policy approach initially explores many states, leading to longer episodes, while off-policy stabilizes quickly, indicating faster convergence in reducing the episode length.

Graph 2: Best Fit Lines for Rewards (R_on and R_off)

Observation: The rewards for both on-policy (blue points) and off-policy (orange points) show variability but tend to cluster around the range of -3 to -4.5. The best-fit lines show a slight upward trend for both on-policy and off-policy rewards. On-policy (black line) has a steeper positive slope

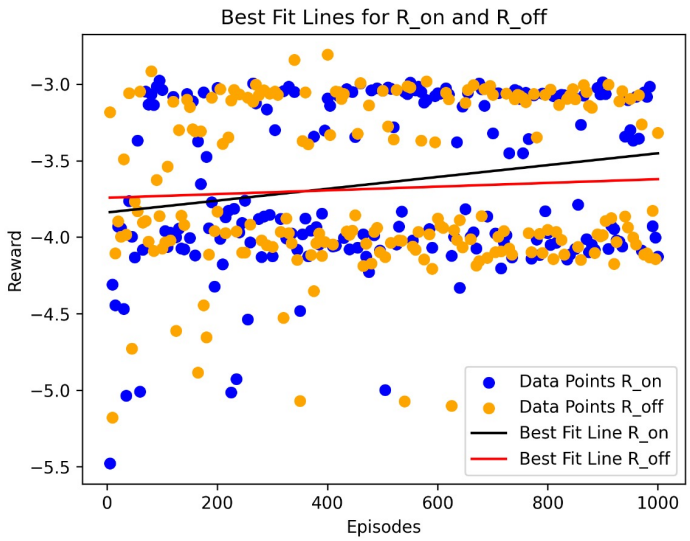

Figure 2: Reward on On and Off policy based on number of episodes

compared to off-policy (red line), indicating that on-policy is improving its reward slightly faster than off-policy.

Analysis: Both methods are learning over time and improving their rewards, but on-policy seems to have a more consistent improvement in maximizing the reward compared to off-policy.

Graph 3: On-Policy vs Off-Policy (V and Q Values)

Observation: Values are only for state (0,0) and action up , only these values have been used because we want to check how fast the 2 policies are converging .Both on-policy (blue) and off-policy (orange) values start off very low (around -6.5) but gradually stabilize around -5.5 to -6 as episodes progress. There is a lot of fluctuation in the values during the early episodes, with both approaches showing similar behavior as they converge.

Analysis: This indicates that both methods are learning a stable policy over time, with on-policy and off-policy methods converging to similar value estimates. While off-policy seems to fluctuate a bit more during convergence, the overall difference in performance between the two seems minimal as the number of episodes increases.

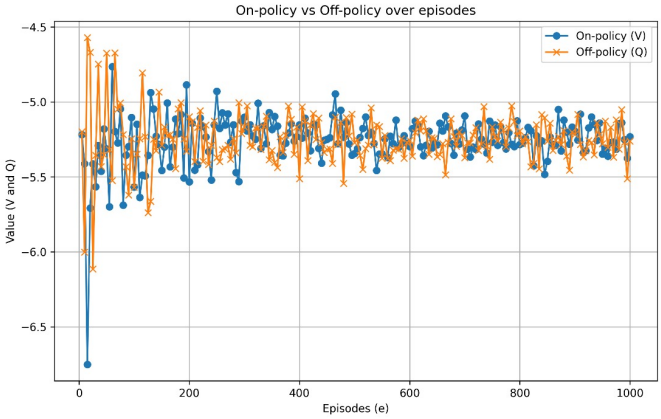

Figure 3: Q and V values for 0,0 and action up

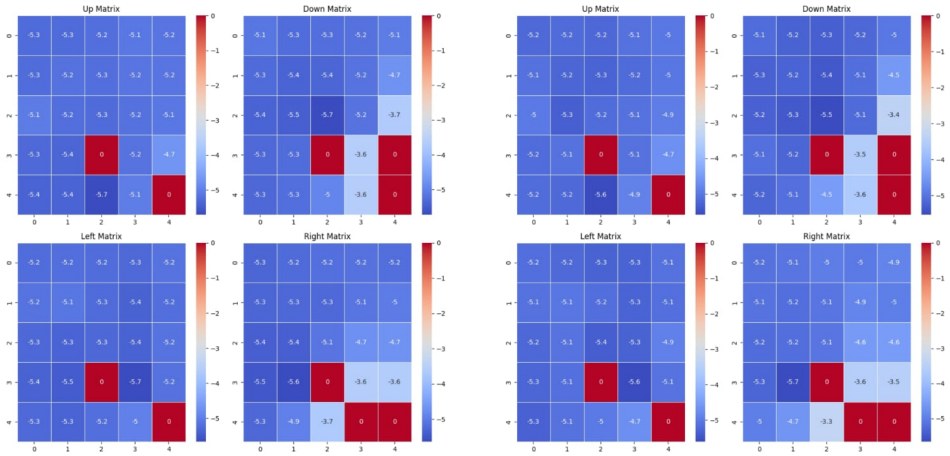

Figure 4: Q values of on and off policy

Summary of On-Policy vs Off-Policy

- **Episode Length:** On-policy starts with longer exploration but stabilizes. Off-policy remains more consistent with shorter episodes, suggesting faster learning initially.
- **Rewards:** Both methods show improvements, with on-policy slightly outperforming off-policy in terms of reward accumulation over time.
- **Value (V and Q):** Both methods converge to similar value estimates, though off-policy exhibits more fluctuation during early episodes.

Conclusion: Both on-policy and off-policy approaches are effective but differ in exploration strategies and speed of convergence. On-policy seems to explore more initially, potentially leading to better rewards in the long run, while off-policy stabilizes faster with fewer fluctuations.