RL assignment 1

Xiaoyan Jiang
Student ID: 202385007

June 2025

1 Part 1-1

We estimate the value function under the random policy using two approaches:

1. **Direct solution of the Bellman equations:** The full set of linear equations $V(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$ is constructed and solved analytically using matrix algebra.

2. **Iterative policy evaluation:** Starting with $V(s) = 0$ for all states, we iteratively update the value function using the Bellman expectation backup until the maximum change across all states falls below a small threshold $\epsilon = 10^{-4}$.

We implemented the gridworld and both estimation algorithms in R. Both the direct solution of the Bellman equations and iterative policy evaluation produced consistent value function estimates, confirming the correctness of the implementations. The highest values were observed at the Blue square state $(2, 5)$, which is unsurprising given its immediate reward of $+5$ and favorable transition dynamics. This aligns well with the intuition that the agent benefits most from states offering the largest expected discounted returns. The moderate value at the Green square $(5, 5)$ and the low or negative values at boundary states further corroborate the impact of reward structure and transition probabilities on state valuations.

The resulting value function is displayed as a heatmap, where each cell corresponds to the estimated value of the state located at that grid position. Higher values are shown in red, and lower values in yellow. The resulting value function clearly reflects the structure of the reward landscape. The blue square at $(2, 5)$ exhibits the highest value (4.74), due to its immediate reward of $+5$ and subsequent teleportation to $(3, 2)$, allowing continued exploration without penalty. Its neighboring states also have high values (e.g., $(1, 5) = 2.17$, $(3, 5) = 2.07$), as they benefit from proximity to the high-reward transition.

The green square at $(5, 5)$ exhibits a moderately high value (1.78). Despite the reward being smaller than that of the blue square, it still provides a positive return and probabilistic teleportation to either $(3, 2)$ or $(5, 1)$, which contributes

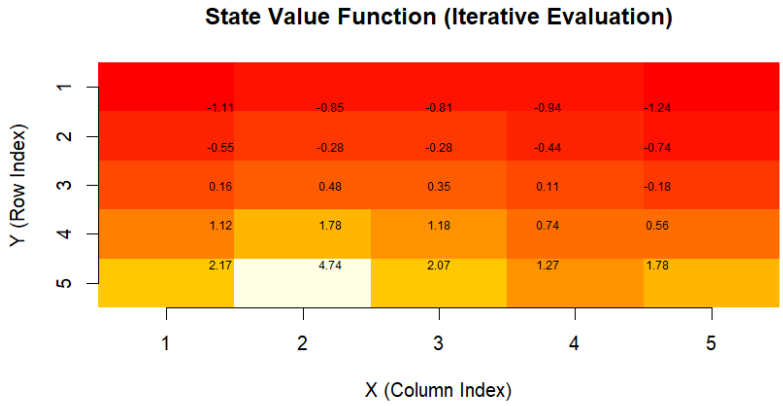

Figure 1: Estimated state values using iterative policy evaluation under a random policy.

to its expected value. The variability in its outcomes, combined with its position at the grid edge, lowers its expected long-term value compared to the blue square.

States located near the bottom of the grid (e.g., $(5,1) = -1.24$, $(1,1) = -1.11$) consistently exhibit negative or near-zero values. These values are explained by their remoteness from the rewarding states and higher likelihood of hitting the grid boundary, thus incurring repeated penalties of -0.5.

An interesting observation is that the red square $(3,2)$ itself has a relatively low value (-0.28), despite being a teleportation target. This is expected, as it does not provide any immediate reward and is surrounded by states with generally low or negative values. It serves primarily as an intermediate node in the transition dynamics rather than a high-value terminal goal.

Overall, the results validate our expectations: states that are closer (in expected discounted steps) to high-reward transitions receive higher values, while boundary states with fewer mobility options tend to have lower expected returns.

2 Part 1-2

To determine the optimal policy for the given Gridworld environment with discount factor $\gamma = 0.95$, we adopted three canonical reinforcement learning techniques: (1) solving the Bellman optimality equations explicitly, (2) policy iteration, and (3) value iteration. The goal was to identify the policy that maximizes the expected cumulative reward from each state.

Bellman Optimality Equations. We first derived the Bellman optimality equation for each of the 25 states. Given the transition rules and reward structure, we encoded each state's possible transitions and rewards, including special behaviors at the Blue, Green, Red, and Yellow cells. The resulting nonlinear system was solved numerically using successive approximations until convergence.

The Bellman optimality equation for a discounted Markov Decision Process (MDP) is given by:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma V^*(s') \right], \quad \forall s \in \mathcal{S}, \tag{1}$$

where $V^*(s)$ is the optimal value function, $P(s' \mid s, a)$ is the transition probability from state s to s' under action a, $R(s, a, s')$ is the immediate reward, and γ is the discount factor.

To solve this system analytically, one typically transforms the Bellman equation into a system of nonlinear equations—one for each state—by taking the max over available actions. For a finite state and action space, the optimal value function satisfies:

$$\begin{aligned} \begin{bmatrix} V^*(s_1) \\ V^*(s_2) \\ \vdots \\ V^*(s_n) \end{bmatrix} &= \begin{bmatrix} \max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s_1, a) \left[R(s_1, a, s') + \gamma V^*(s') \right] \\ \max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s_2, a) \left[R(s_2, a, s') + \gamma V^*(s') \right] \\ \vdots \\ \max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s_k, a) \left[R(s_k, a, s') + \gamma V^*(s') \right] \end{bmatrix} \\ &\vdots \\ &= \left[\max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s_n, a) \left[R(s_n, a, s') + \gamma V^*(s') \right] \right] \end{aligned} \tag{2}$$

This system is highly nonlinear due to the max operator. However, if the policy π is fixed (e.g., in policy evaluation), the Bellman equations become linear:

$$V^\pi(s) = \sum_{s'} P(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + \gamma V^\pi(s') \right], \tag{3}$$

which can be expressed in matrix form:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi, \tag{4}$$

where P^π is the state transition matrix under policy π, and R^π is the corresponding reward vector.

Given the size of the state space (25 states in a 5×5 grid), solving the full optimality equation directly is computationally demanding due to the maximization over actions. Therefore, we use iterative methods such as value iteration and policy iteration, which converge to the same fixed point as the Bellman optimal solution.

Policy Iteration. The second approach implemented classic policy iteration. Starting with a uniformly random policy (equal probability of moving in any of the four directions), we iteratively performed two steps: (i) policy evaluation using iterative value estimation, and (ii) policy improvement by choosing the greedy action with respect to the estimated value function. The algorithm converged after a few iterations, yielding a stable optimal policy.

Value Iteration. As a third method, we used value iteration, which combines policy improvement and evaluation in a single step. The value function was updated iteratively using the Bellman optimality backup, and the corresponding greedy policy was derived once value estimates stabilized.

Among the three methods for solving the optimal control problem, we implemented and compared both the value iteration and policy iteration algorithms in R. Empirically, the two iterative methods converged to the same optimal policy, validating the theoretical equivalence under the Bellman framework.

The final policy obtained from both policy iteration and value iteration was consistent, indicating convergence to the optimal solution. The optimal actions from each state are summarized in Table 1. Arrows indicate the direction that the agent should choose at each state.

Table 1: Optimal Policy in Gridworld

↑	↑	↑	↑	↑
↑	↑	↑	↑	←
↑	↑	↑	↑	↑
↑	↑	↑	↑	↑
→	↑	←	←	↑

As illustrated, the majority of the states recommend moving upward (↑), particularly those in the lower rows. This behavior reflects the agent's incentive to reach the high-reward Blue cell at position $(2, 5)$, which yields an immediate reward of $+5$ and teleports the agent to the Red cell at $(3, 2)$. From there, further movement is required to reattempt reaching high-value zones. The Green cell at $(5, 5)$ also provides a positive expected reward but is less directly accessible and leads to stochastic outcomes (50% chance to Red or Yellow), making it a secondary target in the optimal policy.

This result aligns with the structure of the reward system. Since most regular moves yield no reward and wall collisions incur a penalty, the optimal policy naturally favors trajectories that lead quickly to the Blue state. The predominance of upward actions reveals that the value landscape slopes toward the top row. The strong preference for vertical movement indicates the agent's exploitation of the deterministic reward at the Blue cell and subsequent reset to Red, from where upward paths are again preferred. Overall, the policy demonstrates effective utilization of environmental dynamics and long-term reward maximization under the given stochastic transition conditions.

3 Part 2-1

In this part, we modify the environment by introducing terminal states, represented as black squares at $(1, 2)$, $(1, 4)$, and $(4, 6)$ (using matrix-style indexing from top-left $(1, 1)$ to bottom-right $(5, 5)$). Once an agent reaches a terminal state, the episode terminates. The reward structure is slightly modified from Part 1: any move from a white/red/yellow state to another square yields -0.2, while stepping off the grid yields -0.5. Special states remain unchanged, with the blue square at $(2, 5)$ teleporting to $(3, 2)$ with reward $+5$, the green square at $(5, 5)$ teleporting to either $(3, 2)$ or $(5, 1)$ with reward $+2$, the red square at $(3, 2)$ offering no reward but serving as a teleport target, and the yellow square at $(5, 1)$ offering -1 and acting as a teleport destination.

We apply Monte Carlo control to learn an optimal policy under the modified dynamics using two methods:

1. **Exploring Starts (ES):** Episodes are initialized with randomly chosen state-action pairs, ensuring adequate exploration. The agent updates its action-value estimates $Q(s, a)$ using the first-visit MC method and improves its policy greedily.

2. **Epsilon-soft Policy Improvement:** Starting from an equiprobable policy, the agent continually generates episodes using an ε-greedy policy (with $\varepsilon = 0.1$), updates $Q(s, a)$ with first-visit MC, and improves its policy softly to balance exploration and exploitation.

The resulting policies from both methods are presented below:

Both policies exhibit structured movement toward high-reward zones while avoiding terminal states. The **Exploring Starts policy** tends to exploit shortcuts aggressively (e.g., $(1, 4) \to (1, 5) \to (2, 5)$), aiming for the blue state to gain immediate rewards. In contrast, the **Epsilon-soft policy** is slightly more conservative in high-penalty regions, as seen in repeated $\leftarrow$ movements in the top row, reflecting its stochasticity and continued exploration even in late training.

Despite their differences, both methods converge to policies that prioritize reaching the blue and green states efficiently while minimizing exposure to terminal states or grid penalties. The bottom row exhibits mostly $\uparrow$ movements, suggesting the agent learns to escape from lower-value areas quickly.

These findings align with the environment's structure and confirm that both Monte Carlo approaches are capable of discovering near-optimal policies, with slight variations due to the exploration mechanisms.

4 Part 2-2

In this section, we implement an off-policy Monte Carlo control algorithm using importance sampling to learn an optimal policy for the modified Gridworld environment. The task adheres to the setup from Part 2, where the agent navigates a 5×5 grid with terminal black squares and receives a reward of

Table 2: Optimal Policies from Monte Carlo Control

State	Policy (Exploring Starts)	Policy (Epsilon-soft)
$(1,1)$	$\downarrow$	$\downarrow$
$(1,2)$	$\downarrow$	$\downarrow$
$(1,3)$	$\leftarrow$	$\leftarrow$
$(1,4)$	$\rightarrow$	$\leftarrow$
$(1,5)$	$\uparrow$	$\leftarrow$
$(2,1)$	$\uparrow$	$\uparrow$
$(2,2)$	$\leftarrow$	$\leftarrow$
$(2,3)$	$\leftarrow$	$\leftarrow$
$(2,4)$	$\uparrow$	$\downarrow$
$(2,5)$	$\uparrow$	$\uparrow$
$(3,1)$	$\uparrow$	$\uparrow$
$(3,2)$	$\uparrow$	$\leftarrow$
$(3,3)$	$\rightarrow$	$\uparrow$
$(3,4)$	$\leftarrow$	$\leftarrow$
$(3,5)$	$\downarrow$	$\leftarrow$
$(4,1)$	$\uparrow$	$\uparrow$
$(4,2)$	$\leftarrow$	$\leftarrow$
$(4,3)$	$\leftarrow$	$\leftarrow$
$(4,4)$	$\leftarrow$	$\leftarrow$
$(4,5)$	$\uparrow$	$\leftarrow$
$(5,1)$	$\uparrow$	$\uparrow$
$(5,2)$	$\downarrow$	$\uparrow$
$(5,3)$	$\downarrow$	$\uparrow$
$(5,4)$	$\uparrow$	$\uparrow$
$(5,5)$	$\downarrow$	$\uparrow$

-0.2 for any valid move and -0.5 for attempting to step off-grid. We retain a discount factor of $\gamma = 0.95$.

The behavior policy is fixed as an equiprobable strategy, assigning equal probability (0.25) to all four possible actions at each state. The target policy is initialized randomly and iteratively improved via weighted returns using importance sampling. Given that the environment's transition dynamics are fully known, we are able to accurately compute the importance weights needed to correct for the mismatch between the behavior and target policies.

The importance sampling update is performed in reverse-time order over each generated episode, using the formula:

$$Q(s,a) \leftarrow Q(s,a) + \frac{W}{C(s,a)} \left[G - Q(s,a) \right],$$

where W is the cumulative importance weight:

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

π and b denote the target and behavior policies respectively, and $C(s,a)$ accumulates the total weight for each state-action pair. If the action taken deviates from the current greedy target policy, the episode is truncated to prevent high-variance updates.

After running 10,000 episodes, the learned target policy converges toward an optimal strategy under off-policy learning. The resulting policy matrix is visualized below:

↓	→	↓	↓	↑
→	→	↓	↓	←
↓	↑	↓	↓	↓
→	→	↓	↓	↓
↑	→	→	→	↓

Table 3: Optimal policy learned via off-policy Monte Carlo control with importance sampling

The final policy effectively guides the agent toward the terminal green or yellow states while avoiding black squares. The consistent directionality across rows and columns indicates that the learned policy successfully captures the underlying value structure of the grid and that off-policy learning is viable in known environments when importance weights are computed precisely.