Reinforcement Learning Project 2: Grid world Environment and Monte Carlo Methods

Yeganeh Safari

Mahtab Mohammadi

Introduction

In this project, we explored a 5x5 gridworld environment to understand and compare different reinforcement learning methods for finding optimal policies. The grid consists of 25 states, where an agent can move up, down, left, or right. Special states (blue and green squares) offer different rewards and transitions, influencing the agent's policy and value function.

Objective

1. Study the gridworld problem using different methods to determine the optimal policy.
2. Implement and compare Monte Carlo methods with exploring starts and ε-soft approach.
3. Analyze the effect of random transitions in a dynamic gridworld environment.

Part 1: Gridworld Environment

Bellman Equation

To determine the value of each state in our system, we used the Bellman equations. This involved setting up a series of linear equations where we represented each state's value in a vector. We built a matrix from the probabilities of moving between states and a vector from the rewards we receive. By solving these equations, we could calculate how valuable each state is, taking into account both the immediate rewards and the potential future benefits. This approach gives us a clear understanding of the value of being in each state, helping us make better decisions based on these values.

Results

```
[[ 2.171    4.73362  2.07028  1.26529  1.77912]
 [ 1.11807  1.78212  1.1741   0.73917  0.56247]
 [ 0.16279  0.47789  0.35198  0.11046 -0.18617]
 [-0.54699 -0.28473 -0.2804  -0.43991 -0.74431]
 [-1.10788 -0.84937 -0.80799 -0.93799 -1.23723]]
```

Highest Value: 4.73362 at state (0, 1).

Iterative Policy Evaluation

In Iterative Policy Evaluation, we repeatedly update our estimates of how valuable each state is, adjusting them until they stabilize. This process assumes that every action has an equal chance of being chosen, meaning our policy doesn't favor any particular action over another. By continuously refining these value estimates, we eventually reach a point where the values stop changing significantly, giving us a reliable measure of the value of each state based on the current policy.

Results

```
[[ 2.17121524  4.73386771  2.20154156  1.52050078  2.29226024]
 [ 1.16828339  1.84708998  1.2814004   0.91386428  0.82177337]
 [ 0.20933706  0.53510418  0.43270685  0.22401098 -0.04164667]
 [-0.50526412 -0.23623413 -0.21875609 -0.36188088 -0.65333567]
 [-1.06922357 -0.8059231  -0.75583246 -0.8757868  -1.16786543]]
```

Highest Value: 4.73386771 at state (0, 1).

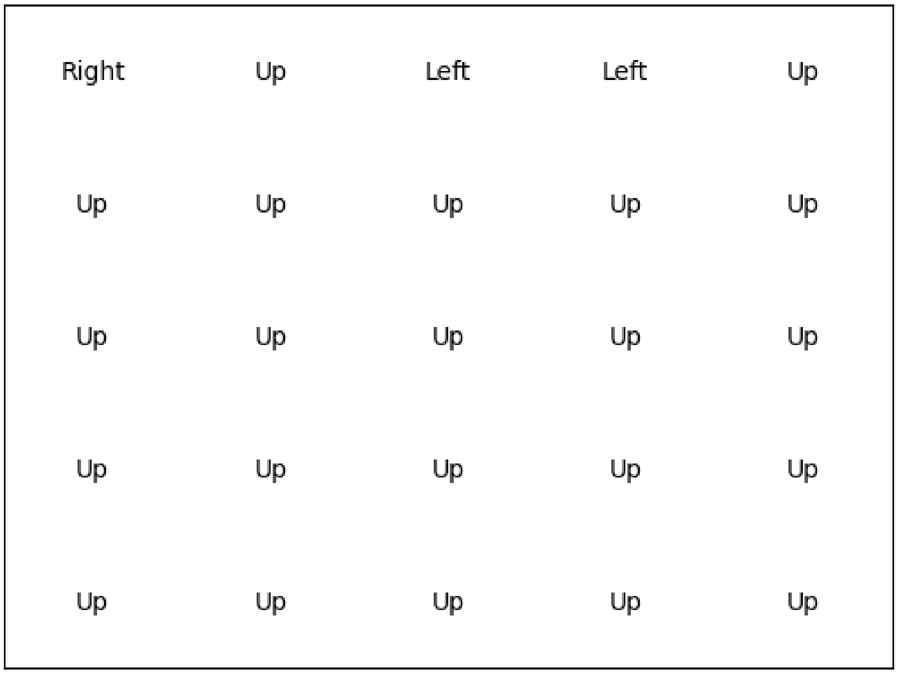
Value Iteration

This algorithm improves the value function by updating the value estimates step by step using the Bellman optimality equation. Essentially, it refines our understanding of the value of each state based on the best possible actions. The results of this process show a value function matrix where each number represents the estimated value of a state. In the final results, the highest value of 22.10 is found at the state position (0, 1), indicating this state is the most valuable according to our updated estimates.

Results

```
[[20.99700533 22.10211088 20.99700533 19.94706036 19.60211088]

 [19.94706036 20.99700533 19.94706036 18.94970734 18.62200533]

 [18.94970734 19.94706036 18.94970734 18.00222198 17.69081036]

 [18.00222198 18.94970734 18.00222198 17.10211088 16.80626984]

 [17.10211088 18.00222198 17.10211088 16.24700533 15.96595635]]
```

Highest Value: 22.10211088 at state (0, 1)



Right	Up	Left	Left	Up
Up	Up	Up	Up	Up
Up	Up	Up	Up	Up
Up	Up	Up	Up	Up
Up	Up	Up	Up	Up

Part 2: Monte Carlo Methods

In Part 2, we examined how our agent performs in a dynamic environment where some states are terminal. We used two different Monte Carlo methods to see how they affect our results.

Monte Carlo Exploring Starts (MC-ES): This method involved starting each episode from a random state and action to get a variety of experiences. The results showed that the agent preferred states that offer higher immediate rewards. For example, the state at position (0,1) had the highest value of 4.65, indicating that it's more rewarding compared to others.

Monte Carlo ε-soft Approach: In this method, we used an ε-soft policy that allows the agent to explore all possible actions, even the less promising ones. This approach gave us a more balanced view of the state values. The results showed different values from the MC-ES method, reflecting the broader exploration.

In summary, while the MC-ES method highlighted a preference for states with higher rewards, the ε-soft approach provided a more thorough exploration of the state space, resulting in a more even distribution of values and policies.

Results

Optimal Value Function (MC-ES):

```
[[ 4.31518378  4.64756188  4.31518378  3.99942459  2.31795497]
 [ 1.33432798  1.51505363  1.70556363  1.63091452  2.10155591]
 [-0.3709875  -0.28525    -0.195      -0.1         0.        ]
 [-0.1        -0.45243813 -0.41582324 -0.1        -0.1       ]
 [ 0.         -0.1        -0.31308782 -0.20892514 -0.1       ]]
```

Optimal Policy (MC-ES):

```
(0, 0): (0, 1)

(0, 1): (-1, 0)

(0, 2): (0, -1)

(0, 3): (0, -1)

(0, 4): (0, -1)
```

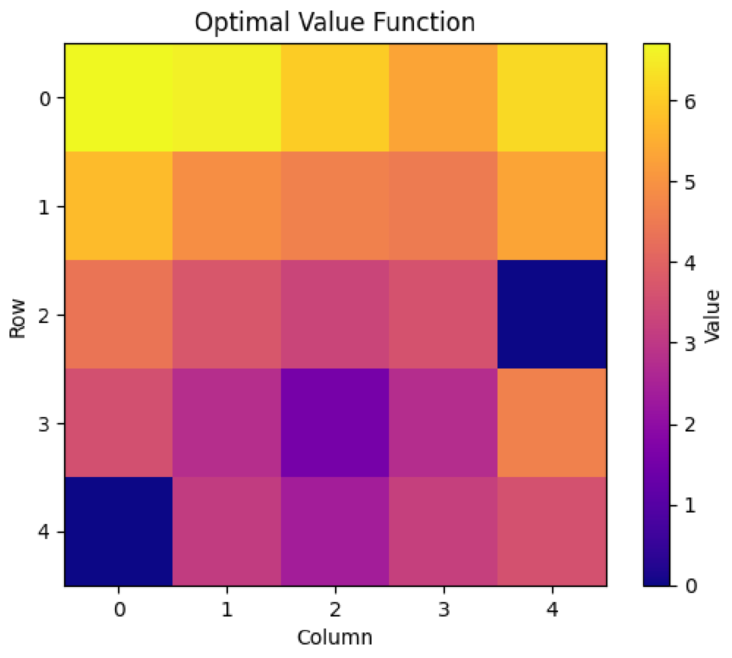
(1, 0): (0, 1)

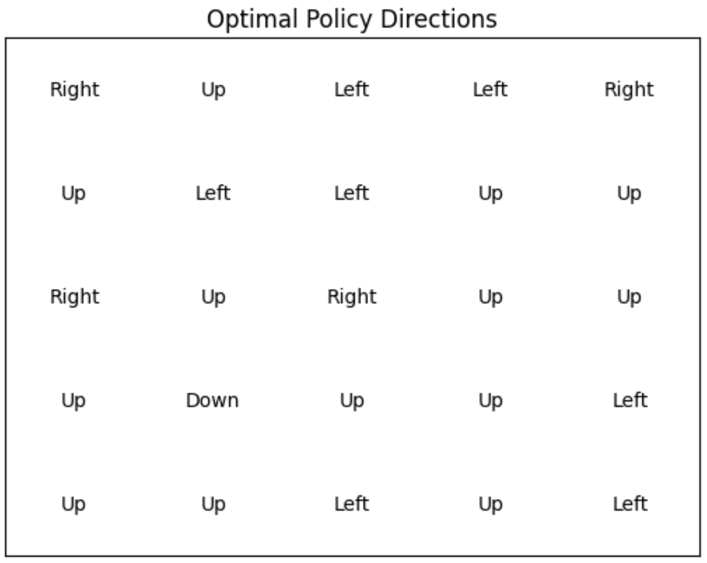
(1, 1): (0, 1)

(1, 2): (0, 1)

(1, 3): (0, 1)

(1, 4): (-1, 0)




Optimal Policy Directions

Right	Up	Left	Left	Right
Up	Left	Left	Up	Up
Right	Up	Right	Up	Up
Up	Down	Up	Up	Left
Up	Up	Left	Up	Left

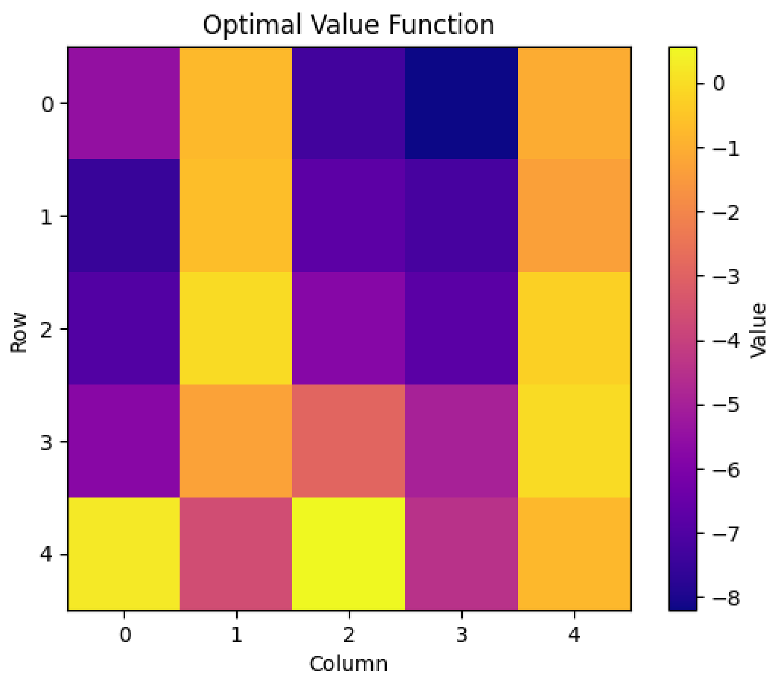
Part 3: Random Transitions

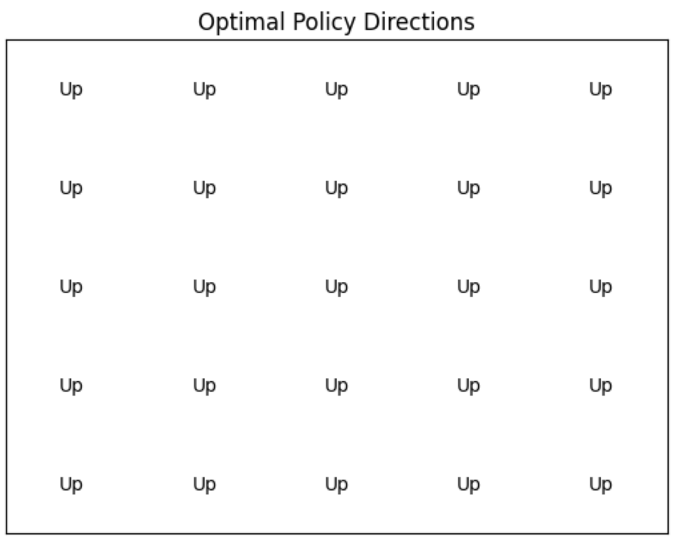
In Part 3 of the project, we introduced randomness to the gridworld environment. Specifically, we added a 10% probability that the positions of the blue and green squares would change at each step. This added complexity requires the agent to adapt its policy to handle the dynamic nature of the environment.

We used the Monte Carlo method to learn the optimal policy in this random environment. At each step, there's a 10% chance that the green and blue squares will randomly change positions within the grid. If the agent steps into the blue square, it gets a reward of 5 and moves to the red square. The green square gives a reward of 2.5 and moves to either the yellow or red square. Terminal states (2,4) or (4,0) end the episode. Moving out of bounds results in a -0.5 reward and the agent stays in place, while moving between white squares yields -0.2. The agent follows a policy to generate episodes, continuing until it reaches a terminal state or a step limit. After each episode, we compute the returns and update the value function and policy incrementally.

Optimal Value Function with Random Policy (V_Random):

```
[[-5.48616383 -0.76427997 -7.29808298 -8.19671298 -1.08123828]

 [-7.49709949 -0.64897637 -6.74550787 -7.2036089  -1.34424579]

 [-6.95758684 -0.05013278 -5.78205279 -6.76500702 -0.28356781]

 [-5.76203054 -1.31438045 -2.91712294 -4.96916956 -0.03906214]

 [ 0.22235482 -3.62118593  0.55944332 -4.44546208 -0.78579031]]
```

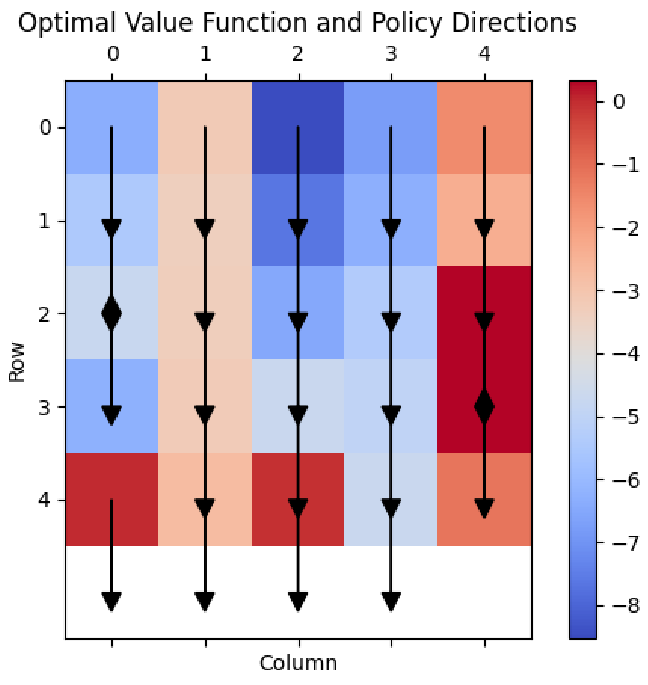




Optimal Value Function with Random Policy (V_Random):

```
[[-6.31480163e+00 -3.22390167e+00 -8.51874866e+00 -6.76114630e+00
  -1.57825481e+00]
 [-5.46863988e+00 -3.37873002e+00 -7.65383813e+00 -6.27986204e+00
  -2.38152700e+00]
 [-4.72511930e+00 -3.33885805e+00 -6.49611773e+00 -5.36476129e+00
```

```
  3.26150124e-01]
 [-6.25003498e+00 -3.24274357e+00 -4.71270800e+00 -4.97031620e+00
  3.02010010e-01]
 [-1.10479963e-07 -2.77172455e+00 -8.38629566e-02 -4.71557379e+00
 -1.18230997e+00]]
```



The heatmap of the Optimal Value Function shows the value of each state, with yellow indicating higher values and blue indicating lower values. The state (3,2) has a high value due to the reward from the blue square. Values decrease as we move away from high-reward states, indicating the agent's preference for these areas. The Policy Directions use arrows to show the optimal move from each state, reflecting the environment's dynamic nature, with a general preference for moving "Up" to maximize rewards. The Combined Visualization integrates the value function and policy directions, illustrating how the agent navigates the grid to maximize cumulative reward.

Conclusion

In this project, we explored different reinforcement learning methods in a gridworld environment. Each method provided insights into the optimal value function and policy directions under various conditions, from fixed transitions to random repositioning of key states. These methods highlight the importance of exploration and adaptation in dynamic environments