Assignment-1: Solving Gridworld

Marks: 80

Gridworld

Gridworld is a classical RL environment with many variants. The following figure displays the visual representation of the environment:

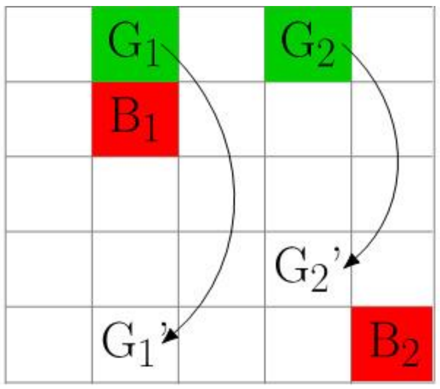

Figure 1: The Gridworld environment

As you can see, the states are represented by cells, and there are 25 states arranged in a 5 x 5 grid. There are four available actions: left, right, up, and down. These actions move the current state in the direction of the action, and the associated reward is 0 for all actions. The exceptions are as follows:

- Border cells: If an action takes the agent outside of the grid, the agent state does not change, and the agent receives a reward of -1.
- Good cells: G1and G2 are good cells. For these cells, each action brings the agent to states and , respectively. The associated reward is +10 for going outside state G1 and +5 for going outside state G2.
- Bad cells: B1 and B2 are bad cells. For these cells, the associated reward is -1 for all actions.

Now that you have an understanding of the environment, let's attempt an activity that implements it.

In this activity, we will be working on the Gridworld environment. The goal of the activity is to calculate and visually represent the state values for a random policy, in which the agent selects each action with an equal probability (1/4) in all states. The discount factor is assumed to be equal to 0.9.

The following steps will help you to complete the activity:

1. Import the required libraries. Import Enum and auto from enum, matplotlib.pyplot, scipy, and numpy, and import tuple from typing.
2. Define the visualization function and the possible actions for the agent.
3. Write a policy class that returns the action probability in a given state; for a random policy, the state can be ignored.
4. Write an Environment class with a step function that returns the next state and the associated reward given the current state and action.
5. Loop for all states and actions and build a transition matrix (width*height, width*height) and a reward matrix of the same dimension. The transition matrix contains the probability of going from one state to another, so the sum of the first axis should be equal to 1 for all rows.
6. Use the matrix form of the Bellman expectation equation to compute the state values for each state. You can use scipy.linalg.solve or directly compute the inverse matrix and solve the system.

The output will be as follows:

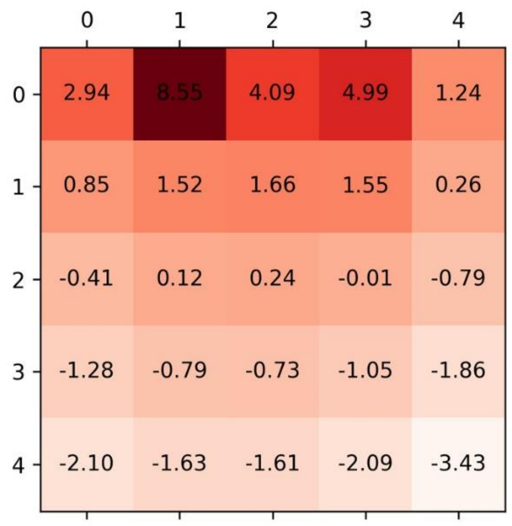

Figure 1: State values of Gridworld

Note:

It is useful to visualize the state values and the expected reward, so write a function visually representing the calculated matrices.