Simple Grid World Environment Tutorial

Your Name

February 16, 2024

1 Introduction

In this tutorial, we will discuss a simple grid world environment that can be used for reinforcement learning. The environment consists of a grid with three states: S1, S2, and a goal state G. The agent can take two actions: Left and Right. The goal of the agent is to reach the goal state G from the starting state S1 while maximizing its total reward.

2 Environment Initialization

Algorithm 1 Initialization of the Simple Grid World Environment

```
observation_space ← Discrete(3)                           ▷ 3 states: S1, S2, G
action_space ← Discrete(2)                                ▷ 2 actions: Left, Right
state ← 0                                                 ▷ Starting state S1
done ← False
Define transition dynamics P and the grid world grid      ▷ See code for details
```

3 Resetting the Environment

Algorithm 2 Resetting the Environment

```
state ← 0                                                 ▷ Reset to starting state S1
done ← False
return state
```

Algorithm 3 Taking a Step in the Environment

transitions $\leftarrow$ P[state][action]
prob, next_state, reward, done $\leftarrow$ transitions[0] $\triangleright$ Assuming deterministic transitions
state $\leftarrow$ next_state
done $\leftarrow$ done
if done **and** reward $\neq 10$ **then**
    reward $\leftarrow -10$ $\triangleright$ Penalize for reaching a non-goal terminal state
end if
return state, reward, done, {}

4 Taking a Step in the Environment

5 Rendering the Environment

The render method is used to visualize the current state of the environment. The grid world is represented as a 2D array where each cell represents a state. The agent's position is indicated by a value of 1 in the grid.

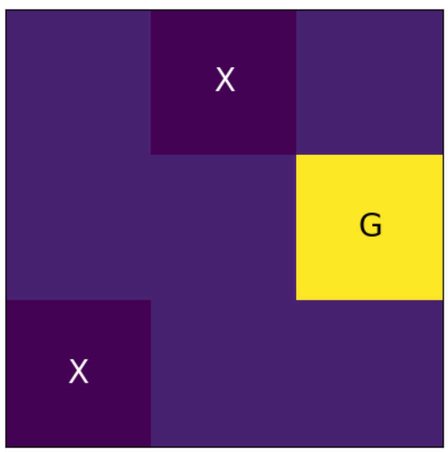

Figure 1: Visualization of the Grid World Environment

6 Implementation

1. create a new notebook 2. import the required library

```
import matplotlib.pyplot as plt
import numpy as np
import gym
```

add a new class

```
class SimpleGridWorldEnv(gym.Env):
    def __init__(self):
        self.observation_space = gym.spaces.Discrete(3)  # 3 states: S1, S2, G
        self.action_space = gym.spaces.Discrete(2)       # 2 actions: Left, Righ
        self.state = 0  # Starting state S1
        self.done = False

        # Define transition dynamics
        self.P = {
            0: {0: [(1.0, 0, -1, False)], 1: [(1.0, 1, -1, False)]},
# Transition from S1
            1: {0: [(1.0, 0, -1, False)], 1: [(1.0, 2, 10, True)]},
# Transition from S2
            2: {0: [], 1: []}  # Terminal state G
        }

        # Define the grid world
        self.grid = np.array([[0, -1, 0], [0, 0, 10], [-1, 0, 0]])

    def reset(self):
        self.state = 0  # Reset to starting state S1
        self.done = False
        return self.state

    def step(self, action):
        transitions = self.P[self.state][action]
        prob, next_state, reward, done = transitions[0]  # Assuming deterministi
        self.state = next_state
        self.done = done
        if done and reward != 10:  # If the agent did not reach the goal state
            reward = -10  # Penalize for reaching a non-goal terminal state
        return self.state, reward, self.done, {}

    def render(self):
        grid_with_agent = np.copy(self.grid)
        grid_with_agent[self.state // 3, self.state % 3] = 1
# Place agent in the grid
        return grid_with_agent
```