DP in Reinforcement Learning

Lidor Erez

June 2024

1 Preface

This is my attempt to solve the gridworld problem using dynamic programming in reinforcement learning.

2 Introduction

Dynamic programming (DP) in reinforcement learning (RL) is a powerful technique that involves breaking down complex problems into simpler sub-problems, solving each sub-problem once, and storing its solution. This method is particularly useful for solving Markov Decision Processes (MDPs), where it helps to determine the optimal policy by iteratively improving the value functions.

However, DP has several limitations that can pose significant challenges when applied to real-life problems. The first major limitation is state space explosion. DP methods require storing and updating a value for every state-action pair. As the number of pairs grows, the computational and memory requirements increase exponentially. For instance, in environments with large or continuous state spaces, the number of possible state-action pairs can be prohibitively large, making DP methods impractical due to their high computational and storage demands.

The second limitation is the necessity for a complete and accurate model of the environment, including the state transition probabilities and the reward function. In many real-world problems, it is not feasible to obtain such a detailed and accurate model because the environment might be too complex, dynamic, or partially observable. This complexity makes it difficult to define precise transition probabilities and rewards.

These limitations underscore the challenges of applying DP in real-world scenarios, driving the need for alternative approaches. Nevertheless, DP remains a valuable tool in certain contexts. In this paper, I will walk through my solution to the gridworld problem using dynamic programming. The goal is to demonstrate that despite its limitations, DP can still be useful and effective in specific cases.

3 Grid World

Consider a simple gridworld where an agent can move in four directions (up, down, left, right). The agent receives a reward of -1 for each move until it reaches a terminal state. Let W be the gridworld matrix of size 4x4 where the bottom-right corner (i=3, j=3) is the terminal state. The goal of the agent is to find the optimal policy that minimizes the total cost to reach the terminal state. The gridworld and the reward for each state is illustrated in Figure 1.

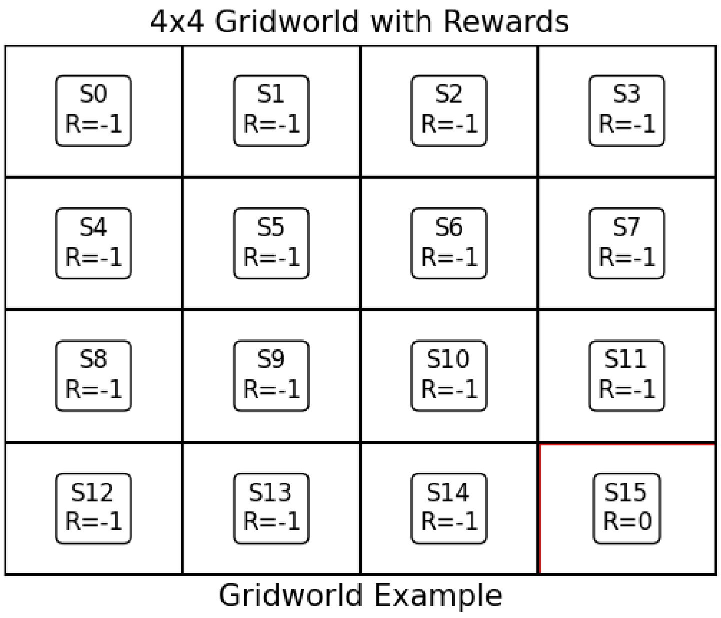

Figure 1: Grid World and rewards for each state

3.1 Solution

To solve this problem, I defined the following parameters:

- $\gamma = 1$
- $\epsilon = 1 \times 10^{-4}$
- Grid size $= 4$
- Terminal state $= (3, 3)$
- $A = \{\text{up, down, left, right}\}$
- Initial state $s_0 = (0, 0)$

Additionally, I have created two functions: policy evaluation and policy improvement that evaluate the state-value function V^π and improve the current policy π_i respectively. The first evaluates V^π for each state until convergence ($|V_k^\pi - V_{k-1}^\pi| < \epsilon$). The evaluation of V^π was done using the formula:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')] \tag{1}$$

The second improves the policy until it becomes greedy (chooses the action that maximizes the reward in the long run). The policy improvement was done using the formula:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\ Q^\pi(s,a) \tag{2}$$

where

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s') \tag{3}$$

Finally, I have used the optimal policy to plot the path taken by the agent.

4 Results

The optimal policy maps the states $\{s_0, s_4, s_8, s_9, s_{10}, s_{14}\}$ to the following actions:

- $\pi(s_0) =$ down leads to s_4
- $\pi(s_4) =$ down leads to s_8
- $\pi(s_8) =$ right leads to s_9
- $\pi(s_9) =$ right leads to s_{10}
- $\pi(s_{10}) =$ down leads to s_{14}
- $\pi(s_{14}) =$ right leads to s_{15}

as illustrated in Figure 2. Recall that s_{15} is the terminal state.

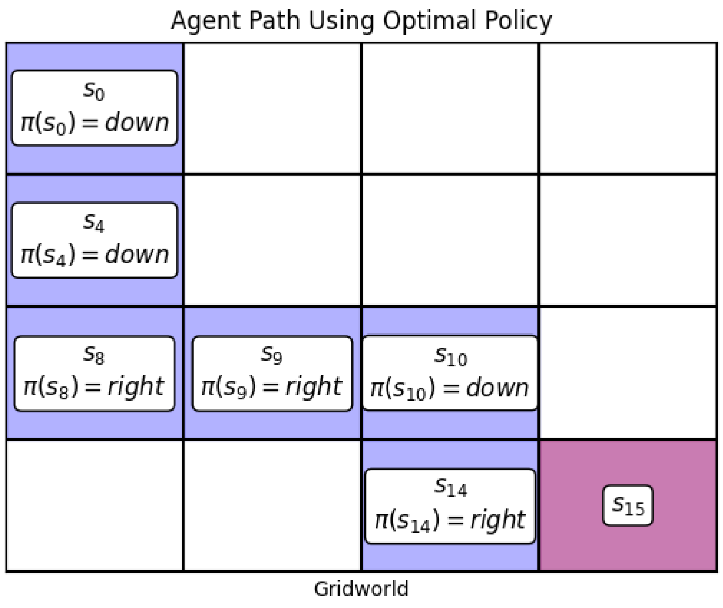

Figure 2: The actions my agent decided to take using the optimal policy

5 Conclusions

In this paper, I demonstrated the application of dynamic programming to solve the gridworld problem. Through policy evaluation and policy improvement, the agent successfully learned the optimal policy to minimize the total cost to reach the terminal state. These results emphasize the power of DP when the model's environment is well specified and the number of states or action-state pairs isn't large.

References

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

[2] Stanford University. (2019). *Reinforcement Learning Course Winter 2019* [YouTube playlist]. YouTube. Retrieved from `https://www.youtube.com/playlist?list=PLoROMvodv4rOSOPzutgyCTapiGlY2Nd8u`