Reinforcement Learning Assignment-3
Markov Decision Process and Dynamic Programming

Utkarsh Prakash
180030042

March 3, 2022

1 Markov Decision Process

A Markov Decision Process is a tuple $< \mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{R} >$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the probability transition function and $\mathcal{R} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the immediate reward function. In order words, $\mathcal{P}(i, j, a) = P_{ij}(a)$ is the probability of transitioning from state i to state j when action a is chosen. Similarly, $\mathcal{R}(i, j, a)$ is the reward obtained on transitioning from state i to state j when action a is chosen.

2 Policy Iteration

Algorithm 1 Policy Iteration

input: MDP $< \mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{R} >$, γ, ϵ
$\mu \leftarrow [0, ..., 0]$ ▷ Initial Policy
$\mu^{'} \leftarrow [0, ..., 0]$ ▷ Temporary for storing present iteration's policy
while $\mu^{'} \neq \mu$ **do**

Policy Evaluation Step
$V_{\mu}^{0} \leftarrow [0, ..., 0]$
$\delta \leftarrow 0$
while $\delta >= \epsilon$ **do**
1. $V_{\mu}^{k+1}(i) = \sum_{j \in \mathcal{S}} P_{ij}(\mu(i))[\mathcal{R}(i, j, \mu(i)) + \gamma V_{\mu}^{k}(j)] \forall i \in \mathcal{S}$
2. $\delta = \max_{i \in \mathcal{S}}(|V_{\mu}^{k+1}(i) - V_{\mu}^{k}(i)|)$

end while

Policy Improvement Step
The value of V_{μ}^{k+1} in the last iteration is V_{μ}. Using this calculate $q_{\mu}(i, a) \forall i \in \mathcal{S}$ and $a \in \mathcal{A}$.
$\mu^{'}(i) = arg \max_{a \in \mathcal{A}} q_{\mu}(i, a) \forall i \in \mathcal{S}$
end while
The policy thus obtained is the optimal policy.

3 Value Iteration

Algorithm 2 Value Iteration

input: MDP $< \mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{R} >$, γ, ϵ
$V^0 \leftarrow [0, ..., 0]$
$\delta \leftarrow 0$
while $\delta >= \epsilon$ **do**
1. $V^{k+1}(i) = \max_{a \in \mathcal{A}} \sum_{j \in \mathcal{S}} P_{ij}(a)[\mathcal{R}(i, j, a) + \gamma V^k(j)] \forall i \in \mathcal{S}$
2. $\mu(i) = arg \max_{a \in \mathcal{A}} \sum_{j \in \mathcal{S}} P_{ij}(a)[\mathcal{R}(i, j, a) + \gamma V^k(j)] \forall i \in \mathcal{S}$
2. $\delta = \max_{i \in \mathcal{S}}(|V^{k+1}(i) - V^k(i)|)$

end while
The policy thus obtained is the optimal policy.

4 Grid World MDP

Let's suppose the agent lives in the 4×3 environment as shown in Table. 1. The reward that the agent gets in a particular state is also indicated in the figure. In each of the state the agent needs to choose an action from {Up, Down, Left, Right}. The agent is successful in reaching the state in which itends to reach by taking an action with probability 0.8 and reaches the states perpendicular to the direction of action with the remaining probability (with both the perpendicular directions being equally-likely). Let's suppose the top-leftmost cell is the origin of a coordinate system. The cell at the coordinate (2, 2) is wall or a prohibited state. The discount factor γ for the MDP is 0.9.

0	0	0	1
0	Wall	0	-100
0	0	0	0

Table 1: Grid World

We run the Policy iteration and Value Iteration algorithms to obtain the optimal policy and optimal value function with $\epsilon = 1e - 10$. We choose the initial policy for Policy Iteration algorithm to be moving in the Up direction for all the states. The results obtained are shown as below:

$\rightarrow$	$\rightarrow$	$\rightarrow$	$\uparrow$
$\uparrow$	Wall	$\leftarrow$	$\leftarrow$
$\uparrow$	$\leftarrow$	$\leftarrow$	$\downarrow$

Table 2: Optimal Policy

5.47	6.31	7.19	8.67
4.80	Wall	3.35	-96.67
4.16	3.65	3.22	1.52

Table 3: Optimal Policy

5 Jack's Car Rental Problem

Example 4.2 from Sutton and Barto [1]: Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited \$10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of \$2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number is n is $\frac{\lambda^n}{n!}e^{-\lambda}$, where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night. The discount factor γ for the MDP is 0.9.

Time Steps: Days
Actions: The net numbers of cars moved between the two locations overnight.
States: The state is the number of cars at each location at the end of the day.

5.1 Policy Iteration

The following graphs show the sequence of policies found by the Policy Iteration algorithm starting with the policy that no car is moved between the two locations overnight and $\epsilon = 1e-4$.

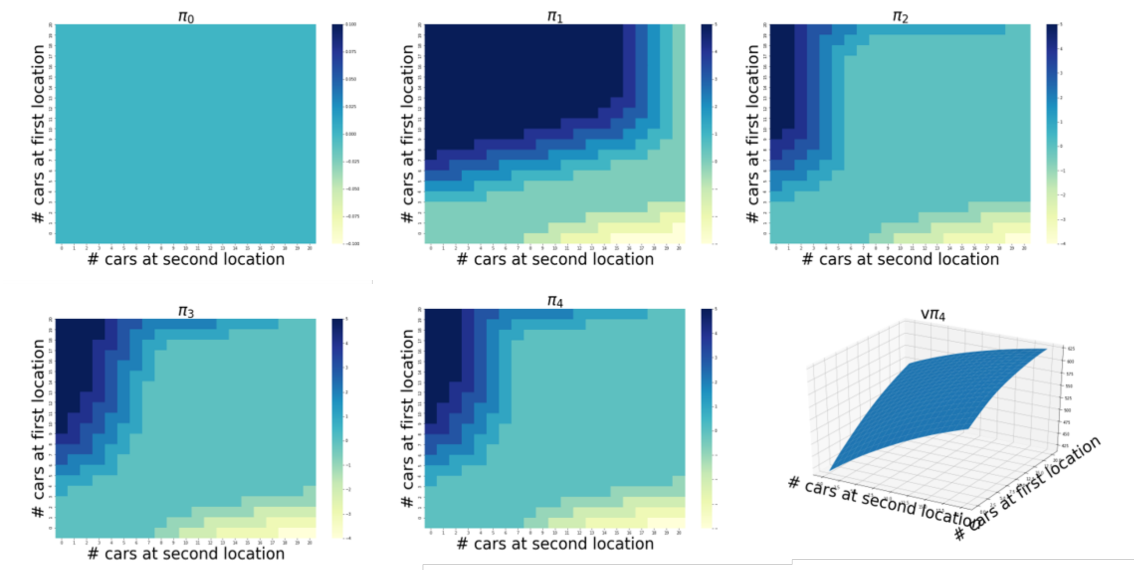

Figure 1

5.2 Value Iteration

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.