Exercise 2

[Avik Banerjee (3374885), Soumyadeep Bhattacharjee (3375428)]

Text in italics are notes taken during the tutorial.

1 Formulating Problems

For large/infinite state spaces, the MDP will be continuous, the state space may be the space of real numbers. However, the state space may be discretized and formulated as a finite MDP (as in the case of formulating 2D space as a grid world). Value or function approximators can be used to discrtize the state space. A grid world can have upper or lower bounds or may be infinite.

a) **Chess:**

- **State space:** All possible combination of legal piece positions on an 8×8 grid make up the state space of chess.
- **Action space:** Set of all legal moves corresponding to each piece.
- **Transitions:** Deterministic.
- **Rewards:** Opponent Wins = -1, Player wins = +1, *0 else*

b) **Pick and Place Robot:**

- **State space:** Working space of the robot divided into finite grids (e.g. 1000×1000). *Camera image can also be used as an observation or for determining the state, but occlusions can cause hindrance. If the camera image is the state space, the states will be Markov states only if a single image is observed at a point of time to determine the next state, multiple states are not stacked.*
- **Action space:** All possible movements of the robot: [Up, Down, Left, Right] + All possible movements of the forklift [Up, Down].
- **Transitions:** Stochastic.
- **Rewards:** Each box placed = +1, Battery runs out before placing 5 boxes = -1, *Continuous reward: distance of object to goal position, discrete reward: object placed in correct position.*

c) **Drone that stabilizes in the air:**

- **State space:** The angle between the arms for the drone and a pre-set perpendicular to the ground (Stable reference angle) can be divided into N States. Based on the current state(or angle) the next action to balance the drone can be chosen.
- **Action space:** All possible movements of the drone: [Up, Down, Left, Right, Diagonal Tilt]. *Control signals for propellers.*
- **Transitions:** Stochastic.
- **Rewards:** Each deviation from the Stable State (reference angle) = -1, Reaching stable state= +1. *Negative reward for velocities, negative reward for crash.*

d) **Self-driving car:**

- **State space:** Total $(M \times N)$ localized state space centered around the car with a grid of $M \times N$.

- **Action space:** All possible movements of the car: [Throttle Up, Throttle Down, Steer Left, Steer Right, Brake, No-Action]
- **Transitions:** Stochastic.
- **Rewards:** Any contact with obstacles = -100, Fuel Runs out / Need to be picked-up = -10, Reaching the destination= +10, Reaching destination before stipulated time = +100.

2 Value Functions

1. A multi-armed bandit has only one state and no transition of state occurs when an action is performed. Each of the arms has a fixed distribution of rewards and the rewards to be acquired in the future does not depend on the present choice of an arm. Hence maximizing the cumulative reward for T trials simply means maximizing the immediate reward at every trial. On the other hand, in an MDP, the future states the agent will traverse depend on the current state and the current action taken by the agent, because of which the expected cumulative reward changes with the policy, whereby future rewards need to be taken into account.

2.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \sum_a \Pr\{A_t = a \mid S_t = s\}\, \mathbb{E}_\pi[G_t \mid S_t = a, A_t = a] \\ &= \sum_a \pi(a \mid s)\, \mathbb{E}_\pi[G_t \mid S_t = a, A_t = a] \\ &= \sum_a \pi(a \mid s)\, q_\pi(s, a) \end{aligned}$$

3.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\big[r + \gamma v_\pi(s')\big] \end{aligned}$$

Now,

$$p(s' \mid s, a) = Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\}$$

and

$$r(s, a, s') = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s']$$

Hence,

$$\sum_{s',r} p(s', r \mid s, a)\, r = \sum_{s'} p(s' \mid s, a)\, r(s, a, s')$$

and

$$\begin{aligned} \sum_{s',r} p(s', r \mid s, a)\, \gamma v_\pi(s') &= \sum_{s'} p(s' \mid s, a) \sum_r p(r \mid s, a, s')\, \gamma v_\pi(s') \\ &= \sum_{s'} p(s' \mid s, a)\, \gamma v_\pi(s') \end{aligned}$$

$\therefore$ the recursive relationship can be defined as

$$\begin{aligned}
v_\pi(s) &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\big[r + \gamma v_\pi(s')\big] \\
&= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\, r + \sum_{s'} \sum_r p(s', r \mid s, a)\, \gamma v_\pi(s') \\
&= \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\, r(s, a, s') + \sum_{s'} p(s' \mid s, a)\, \gamma v_\pi(s') \\
&= \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\, \big[r(s, a, s') + \gamma v_\pi(s')\big]
\end{aligned}$$

3 Bruteforce the Policy Space

a) The number of possible policies is $|\mathcal{A}|^{|\mathcal{S}|}$.

b)

$$\begin{aligned}
v_\pi &= r + \gamma P_\pi v_\pi \\
\Rightarrow \big(I - \gamma P_\pi\big) v_\pi &= r \\
\Rightarrow v_\pi &= \big(I - \gamma P_\pi\big)^{-1} r
\end{aligned}$$

b) The output:

```
Value function for policy_left (always going left):
[0.         0.         0.53691275 0.         0.         1.47651007
 0.         0.         5.        ]

Value function for policy_right (always going right):
[0.41401777 0.77456266 1.31147541 0.36398621 0.8185719  2.29508197
 0.13235862 0.         5.        ]
```

This is entirely expected since going left from every state takes the agent away from the terminal state (goal) and the agent has a lower probability of reaching the goal, which leads to a low cumulative reward. However going right from every state takes the agent closer to the goal and the agent has a higher probability of landing in the goal.

c) The optimal value function v_*:

```
[0.49756712 0.83213812 1.31147541 0.53617147 0.97690441 2.29508197
 0.3063837  0.         5.        ]
```

The number of optimal policies is 32. *For both down and up the probabilities are 1/3 to stay at top-left, 1/3 to go to the right, 1/3 to go down so it does not make any difference.* The optimal actions are as follows:

1. up, right
2. right
3. right

4. down
5. down
6. right
7. left *, because by taking left, the agent will be still in state 7 or will be in the state 4, it won't go into the hole.*
8. any action
9. any action

d) The brute force approach is applicable only for a small solution space. In realistic scenarios, the action and state spaces are too large for brute force to be a feasible approach. *Accurate knowledge of dynamics is assumed.*