Markov Decision Processes **April 22, 2024**

Module 2.B: Finite-Horizon Markov Decision Processes

Lecturer: Avik Kar

1 The problem

Definition 1.1 (Finite-horizon Markov decision process)**.** We refer to the tuple $(\mathcal{X},\mathcal{U},H,p,\bar{r}) =: M$ as a Markov decision process where

1. $\mathcal{X}$ is the state space of the system.
2. $\mathcal{U}$ is the action space. At system state $x \in \mathcal{X}$, the agent can take action from $\mathcal{U}_x \subseteq \mathcal{U}$.
3. $H \in \mathbb{N}$ is the horizon length.
4. Let $(\Omega,\mathcal{F},\mathbb{P})$ be the probability space and $\mathcal{B}_{\mathcal{X}}$ be a collection measurable sets in $\mathcal{X}$. The collection of transition kernels, $\{p_h : \mathcal{X} \times \mathcal{U} \times \mathcal{B}_{\mathcal{X}} \to [0,1] \mid h \in [H]\}$, governs the state transitions as
$$p_h(x,u,B) := \mathbb{P}\left(X_{h+1} \in B \mid X_h = x, U_h = u\right) \; \forall (x,u,B) \in \mathcal{X} \times \mathcal{U} \times \mathcal{B}_{\mathcal{X}}. \tag{1}$$
5. $\{\bar{r}_h : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \mid h \in [H-1]\}$ be the collection of reward maps such that at the decision epoch, $h \in [H-1]$ the agent receives reward $\bar{r}_h(X_h,U_h) =: R_h$. $\bar{r}_H : \mathcal{X} \to \mathbb{R}$ denote the terminal reward function. At the end of the horizon, the agent earns a reward of $\bar{r}_H(X_H) = R_H$.

Assumption 1.2. We assume the following:

1. Both $\mathcal{X}$ and $\mathcal{U}$ are finite.
2. $\bar{r}_h$ for all $h \in [H]$ are bounded.

Definition 1.3. Consider the policy $\pi = \{\phi_h : \mathcal{X} \to \Delta_{\mathcal{U}}, h \in [H-1]\}$. π is called an admissible policy for M if $\phi_h(x)$ has support on $\mathcal{U}_x$ for every $x \in \mathcal{X}$, $h \in [H-1]$.

Definition 1.4. We define the value function of policy, π as

$$\begin{aligned} V^{\pi}(x_0) &\triangleq \mathbb{E}\left[\bar{r}_H(X_H) + \sum_{h=0}^{H-1} \bar{r}_h(X_h,U_h) \middle| X_0 = x_0, U_h \sim \phi_h(X_h) \; \forall h \in [H-1]\right] \\ &= \mathbb{E}_{\pi}\left[\bar{r}_H(X_H) + \sum_{h=0}^{H-1} \bar{r}_h(X_h,U_h) \middle| X_0 = x_0\right]. \end{aligned} \tag{2}$$

The optimal value function $V^* : \mathcal{X} \to \mathbb{R}$ is defined as

$$V^*(x_0) \triangleq \sup_{\pi \in \Pi} V^{\pi}(x_0) \tag{3}$$

We call a policy π^* optimal if it attains V^* for every $x \in \mathcal{X}$, i.e.,

$$V^{\pi^*}(x) = V^*(x) \tag{4}$$

for every $x \in \mathcal{X}$.

Since X_h, U_h are random variables with well-defined distributions, and $\bar{r}_h$ are measurable and bounded functions for every h, V^{π} is a bounded, well-defined function, so is V^*.

2 Dynamic programming

Proposition 2.1. *For every initial state x_0, the optimal value $V^*(x_0)$ of M is equal to $V_0(x_0)$, where the function V_0 is given by the last step of the following algorithm, which proceeds backward in time from period $H-1$ to period 0:*

$$V_H(x_H) = \bar{r}_H(x_H), \tag{5}$$

$$V_h(x_h) = \max_{u \in U_{x_h}} \left\{ \bar{r}_h(x_h, u) + \sum_{y \in \mathcal{X}} p(x_h, u, y) V_{h+1}(y) \right\}, \;\; h \in [H-1]. \tag{6}$$

Furthermore if $u_h^* = \phi_h^*(x_h)$ maximizes the right side of (6) for each x_h and h, then the policy $\pi^* := \{\phi_h^* | h \in [H-1]\}$ is the optimal policy for the problem.

Proof. For any admissible policy $\pi = \{\phi_h | h \in [H-1]\}$ and each $h \in [H-1]$, denote $\pi^h = \{\phi_{h'} | h' \in \{h, \dots, H-1\}\}$ for $h \in [H-1]$. Let $V_h^*(x_h)$ be the optimal value for the $(H-h)$-stage problem that starts at state x_h and time $h \in [H-1]$, and ends at time H that is,

$$V_h^*(x_h) := \max_{\pi^h} \mathbb{E}_{\pi^h} \left[\bar{r}_H(X_H) + \sum_{h'=h}^{H-1} \bar{r}_{h'}(X_{h'}, U_{h'}) \middle| X_h = x_h \right]. \tag{7}$$

By backward induction, we will show that $V_h^* = V_h$ and, in particular, $V^* = V_0^* = V_0$.

For $h = H$, $V_H^*(x_H) = \bar{r}_H(x_H) = V_H(x_H)$ for every $x_H \in \mathcal{X}$. Assume the induction hypothesis, $V_{h+1}^* = V_{h+1}$. We write $\pi^h = (\phi_h, \pi^{h+1})$.

$$\begin{aligned}
V_h^*(x_h) &= \max_{\pi^h} \mathbb{E}_{\pi^h} \left[\bar{r}_H(X_H) + \sum_{h'=h}^{H-1} \bar{r}_{h'}(X_{h'}, U_{h'}) \middle| X_h = x_h \right] \\
&= \max_{\phi_h} \mathbb{E} \left[\bar{r}_h(X_h, \phi_h(X_h)) + \max_{\pi^{h+1}} \mathbb{E}_{\pi^{h+1}} \left[\bar{r}_H(X_H) + \sum_{h'=h+1}^{H-1} \bar{r}_{h'}(X_{h'}, U_{h'}) \middle| X_h = x_h, U_h \sim \phi_h(x_h) \right] \middle| X_h = x_h \right] \\
&= \max_{\phi_h} \mathbb{E} \left[\bar{r}_h(X_h, \phi_h(X_h)) + \mathbb{E} \left[V_{h+1}^*(X_{h+1}) \middle| X_h = x_h, U_h \sim \phi_h(x_h) \right] \middle| X_h = x_h \right] \\
&= \max_{\phi_h} \mathbb{E} \left[\bar{r}_h(X_h, U_h) + V_{h+1}(X_{h+1}) | X_h = x_h, U_h \sim \phi_h(x_h) \right] \qquad \text{[By induction hypothesis]} \\
&= \max_{\phi_h} \left\{ \sum_{u \in U_h} \phi(u \mid x_h) \left(\bar{r}_h(x_h, u) + \sum_{y \in X_h} p(x_h, u, y) V_{h+1}(y) \right) \right\} \\
&= \max_{u \in \mathcal{U}_{x_h}} \left\{ \bar{r}_h(x_h, u) + \sum_{y \in X_h} p(x_h, u, y) V_{h+1}(y) \right\} = V_h(x_h).
\end{aligned} \tag{8}$$

This concludes the induction argument. Let $\pi^{\star, h+1}$ is optimal for the $H-h-1$ step problem, that is

$$V_{h+1}^*(x) = \mathbb{E}_{\pi^{\star, h+1}} \left[\sum_{h'=h+1}^{H-1} \bar{r}_{h'}(X_{h'}, U_{h'}) \middle| X_{h+1} = x_{h+1} \right],$$

for every $x \in \mathcal{X}$. Let us define the decision rule $\phi_h^* : \mathcal{X} \to \mathcal{U}$ such that

$$\phi_h^*(x) \in \arg\max_{u \in \mathcal{U}_x} \left\{ \bar{r}_h(x, u) + \sum_{y \in X_h} p(x, u, y) V_{h+1}^*(y) \right\},$$

and define $\pi^{\star, h} = \{\phi_{h'} \mid h' \in \{h, \dots, H-1\}\}$. Then,

$$V_h^*(x_h) = \mathbb{E}_{\pi^{\star, h}} \left[\bar{r}_H(X_H) + \sum_{h'=h}^{H-1} \bar{r}_{h'}(X_{h'}, U_{h'}) \middle| X_h = x_h \right]. \tag{9}$$

Proceeding backwards like this we get that $\pi^\star = \pi^{\star, 0}$ maximizes $V_0^\star(x)$ for all $x \in \mathcal{X}$. This concludes the proof. □

Here is the dynamic programming algorithm (alternatively, value iteration algorithm) for episodic MDP:

Algorithm 1 Dynamic Programming

```
Input: Episodic MDP, $M$.
Initilize $V_H = \bar{r}_H$, $\pi^\star = \{\}$
**for** $h = H - 1$ to 0 **do**
    **for** $x \in \mathcal{X}$ **do**
        $V_h(x) = \max_{u \in \mathcal{U}} r_h(x,u) + \sum_{y \in \mathcal{X}} p(x,u,y) V_{h+1}(y)$
    **end for**
    $\phi_h^\star(x) \in \arg\max_{u \in \mathcal{U}} r_h(x,u) + \sum_{y \in \mathcal{X}} p(x,u,y) V_{h+1}(y)$
    $\pi^\star \leftarrow \{\phi_h^\star\} \cup \pi^\star$.
**end for**
Return $V_0$, $\pi^\star$
```

3 Examples

Exercise 3.1 (Inventory control)**.** Consider the inventory control example from Module 2.A. We simplify the problem with $H = 3$, $C = 2$, $\bar{r}_3(x_3) = -2x_3$, $\bar{r}_h(x_h, u_h, w_h) = -u_h - (x_h + u_h - w_h)^2$ for $h = 0, 1, 2$. The probability distribution of w_h for all h are as follows.

$$\mathbb{P}(w_h = 0) = 0.1, \quad \mathbb{P}(w_h = 1) = 0.7, \quad \mathbb{P}(w_h = 2) = 0.2.$$

Find the optimal policy and the optimal value for the problem using dynamic programming.

Write a program (preferably in python) to solve the problem considering $H = 10$.

Exercise 3.2 (Machine replacement)**.** Machine replacement is often used as a benchmark problem in MDPs. The state space $\mathcal{X} = \{1, 2, \ldots, N\}$ corresponds to the machine quality. State 1 denotes the worst machine performance, while N corresponds to the best state. The action space is $\mathcal{U} \in \{1\textit{(replacemachine)}, 2\textit{(use existing machine)}\}$. When action $u = 1$ is chosen, the resulting brand-new machine starts at state N. When action $u = 2$ is chosen, the machine state will either remain the same or deteriorate by one unit. Let θ denote the probability that the machine state will deteriorate by one unit.

Suppose the number of states $N = 3$. Then the transition probabilities $P(u), u \in \mathcal{U}$, of the MDP are:

$$P(1) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$
$$P(1) = \begin{bmatrix} 1 & 0 & 0 \\ \theta & 1-\theta & 0 \\ 0 & \theta & 1-\theta \end{bmatrix}.$$

Operating a machine in state x incurs a reward $r(x, u = 2) = r(x)$ due to possible loss of productivity and poor quality of the machine output. Let r be a monotonically increasing function. On the other hand, replacing the machine at any state x costs R. So $r(x, u = 1) = -R$. The aim is to minimize the expected cumulative cost $\mathbb{E}_\pi \left[\sum_{h=0}^{H-1} r(x_h, u_h) \mid x_0 \right]$ for some specified horizon H. Write down the Bellman equation and solve for the optimal value and the optimal policy for $H = 4$, $r : x \mapsto x$.

Exercise 3.3. Assume that Emma wants to open a cheese station at an office canteen. The cheese must be purchased at the counter of this station. The following contains the details of this scenario.

- Every weekday morning, Emma must decide how much cheese to buy from the vendor.
- Each cheese packet has 100, 200, 300, 400 or 500 slices.
- Every workday, she can purchase only one cheese packet or nothing.

- Each cheese slice is estimated to cost 10 rupees.
- Each cheese slice costs 12 rupees to purchase from the mess counter.
- During weekdays, the canteen fridge can only hold 500 of cheese slices.
- However, owing to the weekend cleaning routine, the fridge does not operate on weekends. Thus, there is no cheese counter on Saturdays and Sundays.
- Any cheese slices that haven't been sold by Friday evening will be wasted as they will not be consumable.
- If the number of slices purchased and the number of existing slices exceeds the capacity on a weekday, the extra inventory is wasted.
- The distribution of demand on any given day is given as follows: For 100, 200, 300, 400, and 500 cheese slices, respectively, the corresponding probabilities are 0.15, 0.05, 0.3, 0.25, and 0.25.
- Because Emma can not sell more cheese than the number of slices available, sales in a day are limited by the minimum between the demand and available slices.
- Emma wants to maximize the total expected profit of the cheese counter over the whole week.

Now, given the context, model the problem as a finite horizon MDP.

The environment is given by the CheeseCounter class. In this class, the demand at each of the days is given by demand value, and the corresponding probabilities are given by demand probabilities. On each weekday, the mess master can buy cheese from the following set $\{0, 100, 200, 300, 400, 500\}$ of cheese slices. So, the possible action spaces are given by the action space variable. The state is a tuple of the day of the week (or the weekend) and the starting cheese level for that day, for example ("Tuesday", 100). Next, get the next state reward is a method that calculates the next state and the reward along with the current state, the action, and the demand. The template also has another method called get transition prob that returns all possible transitions and rewards for a given state-action pair using the get next state reward method, together with the corresponding probabilities. Your task is to complete the functions of iterative policy evaluation and value iteration using the algorithm taught in the class. One can refer to the algorithms from chapter 4 of Sutton Barto textbook. The example policy function is the demo policy. Assume all the policies are deterministic.

A policy is given in the form of a dictionary that maps states to action probabilities; for example, Policy = ('Monday', 0): 100: 0.4, 300: 0.6, ('Tuesday', 0): 100: 0.4, 300: 0.6.... And the value of states is given by a dictionary such as Values v = ('Monday', 0): 2884.0, ('Tuesday', 0): 2204.0, ('Tuesday', 100): 3204.0, ...