MRP

천정민

서울과학기술대학교
SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

I. Recap

Motivation

- I drink a bottle of soda everyday. I drink either Coke or Pepsi everyday. When I choose what to drink for today, I only consider what I drank yesterday.
- Specifically,
  - Suppose I drank Coke yesterday, then the chance of drinking Coke again today is 0.7.
  - (What is the chance of drinking Pepsi today then?)
  - Suppose I drank Pepsi yesterday, then the chance of drinking Pepsi again today is 0.5.
  - (What is the chance of drinking Coke today then?)
- Given I drink coke today, what is likely my consumption for upcoming 10 days? (Pepsi is $1 and Coke is $1.5)

- A Markov chain is a stochastic process with the specification of
  - a state space S
  - a transition probability matrix $\mathbf{P}$

- A Markov reward process is a Markov chain with the specification of
  - a reward r_t with the reward function $R(s) = \mathbb{E}[r_t | S_t = s]$
  - a time horizon H, which is the duration we are interested in cumulative sum of rewards.

- The return G_t is the sum of remaining reward at time t.
  - $G_0 = r_0 + r_1 + \cdots + r_9$
  - $G_1 = r_1 + \cdots + r_9$
  - $G_2 = r_2 + \cdots + r_9$
  - $\cdots$
  - $G_9 = r_9$

- A state-value function $V_t(s)$ is the expected return given state s at time t. That is, $V_t(s) = \mathbb{E}[G_t | S_t = s]$

II. Method 1 - Monte-Carlo simulation

MC simulation for estimating state-value function

- Formally, for a finite-horizon MRP, the following is MC simulation for estimating state-value function.

```
# MC evaluation for state-value function
# with state s, time 0, reward r, time-horizon H
episode_i <- 0
cum_sum_G_i <- 0
while episode_i < num_episode
  Generate an stochastic path starting from state s and time 0.
  Calculate return G_i <- sum of rewards from time 0 to time H-1.
  cum_sum_G_i <- cum_sum_G_i + G_i
  episode_i <- episode_i + 1
State-value-fn V_t(s) <- cum_sum_G_i/num_episode
return V_t(s)
```

- Remark that the full stochastic evolution, previously marked as `MC_i` is replaced by the term `episode_i`. Episode refers to a full single stochastic path from now on.

III. Method 2 - Iterative solution

Motivation

- Same as the previous section, our goal is still to estimate $V_0(c) = \mathbb{E}[G_0|S_t = c]$.
- Since $G_t = \sum_{i=t}^{9} r_i$ has less number of terms when t is high number, we shall start from $t = 9$ and work backward, i.e. from $V_9(s)$, then $V_8(s)$, then $V_7(s)$,$\cdots$

- For $t = 9$,
  - From the general formula $V_t(s) = \mathbb{E}[G_t|S_t = s]$, it is easy to see that $V_9(s) = \mathbb{E}[G_9|S_9 = s] = \mathbb{E}[\sum_{i=9}^{9} r_i|S_9 = s] = \mathbb{E}[r_9|S_9 = s] = R(s)$.
  - In other words,
    - $V_9(c) = \mathbb{E}[r_9|S_9 = c] = R(c) = 1.5$ and
    - $V_9(p) = \mathbb{E}[r_9|S_9 = p] = R(p) = 1.0$.
  - In general,
  $$V_9(s) = R(s) + V_{10}(s), \tag{1}$$
  where $V_{10}(s) = 0, \ \forall s$

- For $t = 8$,
  - From the general formula $V_t(s) = \mathbb{E}[G_t|S_t = s]$, (watch below carefully)

$$\begin{aligned} V_8(s) &= \mathbb{E}[G_8|S_8 = s] \\ &= \mathbb{E}\left[\sum_{i=8}^{9} r_i \mid S_8 = s\right] \\ &= \mathbb{E}[r_8 + r_9|S_8 = s] \\ &= \mathbb{E}[r_8|S_8 = s] + \mathbb{E}[r_9|S_8 = s] \\ &= R(s) + \mathbb{E}[r_9|S_8 = s] \end{aligned} \tag{2}$$

  - Here, let's consider $\mathbb{E}[r_9|S_8 = c]$ first.
    - This is expected spending on day-9 given that I drink coke on day-8. This value is conditioned on what I drink on day-9. If coke on day-9 with probability 0.7, $r_9 = 1.5$. If pepsi w/ prob. 0.3, $r_9 = 1.0$. This expectation is 1.35 (= 0.7 · 1.5 + 0.3 · 1.0).
    - Formally, (using $\mathbb{E}(X|A) = \mathbb{E}(X|E_1, A)\mathbb{P}(E_1|A) + \mathbb{E}(X|E_2, A)\mathbb{P}(E_2|A)$)

$$\begin{aligned} & \mathbb{E}[r_9|S_8 = c] \\ = \; & \mathbb{E}[r_9|S_9 = c, S_8 = c]\mathbb{P}(S_9 = c|S_8 = c) + \mathbb{E}[r_9|S_9 = p, S_8 = c]\mathbb{P}(S_9 = p|S_8 = c) \\ = \; & \mathbf{P}_{cc}\mathbb{E}[r_9|S_9 = c] + \mathbf{P}_{cp}\mathbb{E}[r_9|S_9 = p] \; (\because \text{ Markov property}) \\ = \; & \mathbf{P}_{cc}\mathbb{E}[G_9|S_9 = c] + \mathbf{P}_{cp}\mathbb{E}[G_9|S_9 = p] = \mathbf{P}_{cc}V_9(c) + \mathbf{P}_{cp}V_9(p) \end{aligned}$$

- (Cont'd for $t = 8$)
  - Now, let's consider $\mathbb{E}[r_9|S_8 = s]$ for the generalized state s. With the notation assuming a transition from this state s to the next state s',

$$\begin{aligned}\mathbb{E}[r_9|S_8 = s] &= \mathbf{P}_{sc}V_9(c) + \mathbf{P}_{sp}V_9(p) \\ &= \sum_{s' \in S} \mathbf{P}_{ss'}V_9(s') \qquad (3)\end{aligned}$$

  - We shall now summarize for $t = 8$,

$$\begin{aligned}V_8(s) &= \mathbb{E}[G_8|S_8 = s] = \mathbb{E}[r_8 + G_9|S_8 = s] \\ &= R(s) + \mathbb{E}[G_9|S_8 = s] \\ &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'}V_9(s') \qquad (4) \\ \text{(expected return at time 8)} &= \text{(reward at time 8)} + \text{(expected return at time 9)}\end{aligned}$$

- For $t = 7$,
  - From the general formula $V_t(s) = \mathbb{E}[G_t | S_t = s]$,

$$\begin{aligned} V_7(s) &= \mathbb{E}[G_7 | S_7 = s] \\ &= \mathbb{E}\left[\sum_{i=7}^{9} r_i \mid S_7 = s\right] \\ &= \mathbb{E}[r_7 + r_8 + r_9 | S_7 = s] \\ &= \mathbb{E}[r_7 | S_7 = s] + \mathbb{E}[r_8 + r_9 | S_7 = s] \\ &= R(s) + \mathbb{E}[G_8 | S_7 = s] \end{aligned} \tag{5}$$

  - You got the hint? From here, we want to use $V_8(s) = \mathbb{E}[G_8 | S_8 = s]$ to express this as a recursive formula for state-value function just like Eq. (4).

$$\begin{aligned} V_7(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} \mathbb{E}[G_8 | S_7 = s, S_8 = s'] \\ &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_8(s') \end{aligned} \tag{6}$$

- So far,

$$
\begin{aligned}
V_{10}(s) &= 0 \\
V_9(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_{10}(s') \text{ from Eq. (1)} \\
V_8(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_9(s') \text{ from Eq. (4)} \\
V_7(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_8(s') \text{ from Eq. (6)} \\
\cdots &= \cdots \\
V_t(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_{t+1}(s') \\
\cdots &= \cdots \\
V_0(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_1(s')
\end{aligned}
$$

- Note that the array of equations can be solve from the top to the bottom.
- This iterative method is called as backward induction that works well with finite horizon problem.
- This iterative method (and its painful derivaion) is the most important mathematical essence of Markov decision process.

Implementation strategy

- Summary so far

$$
\begin{aligned}
V_{10}(s) &= 0 \\
V_t(s) &= R(s) + \sum_{s' \in S} \mathbf{P}_{ss'} V_{t+1}(s') \text{ (for } t \in \{0, 1, ..., 9\})
\end{aligned}
$$

- Strategy
  - Column vector v_t for $V_t(s)$
  - Column vector R for $R(s)$
  - The term $\sum_{s' \in S} \mathbf{P}_{ss'} V_{t+1}(s')$ can be written as $\mathbf{P} v_{t+1}$.
  - It follows

$$v_t = R + \mathbf{P} v_{t+1},$$

simply a system of linear equations!

```python
import numpy as np

# Transition matrix P (R fills by column; equivalent Python layout shown)
P = np.array([[0.7, 0.3],
              [0.5, 0.5]], dtype=float)
# Reward vector R
R = np.array([[1.5],
              [1.0]], dtype=float)
# Time-horizon
H = 10
# v_{t+1} initialization (terminal value is zero)
v_t1 = np.zeros((2, 1), dtype=float)

t = H - 1
while t >= 0:
    # Bellman recursion: v_t = R + P * v_{t+1}
    v_t = R + P @ v_t1
    # step backward
    t -= 1
    # shift for next iteration
    v_t1 = v_t

# v_t now holds v_0(c), v_0(p) as a column vector
print(v_t)
```

```
## [[13.35937498]
##  [12.73437504]]
```

```
"0813-Jeongmin"
```