Lecture D1. Markov Reward Process 1

Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr

서울과학기술대학교 데이터사이언스학과

I. Motivation

Going forward

- The learning process is very cumulative.
- This part (Part. D) contains MRP and DP, in the preparation for MDP.

Recap

- In the first introduction of soda DTMC, the following question was posed.

  Given I drink coke today, what is likely my consumption for upcoming 10 days? (Pepsi is \$1 and Coke is \$1.5)

- In Lecture note `C1, Sec. 4`, we demonstrated Monte-Carlo method that generates 10,000 (`MC_N`) number of paths and found total expected cost to be approximately 13.36.

- This lecture builds more systematic approach rather than the previous time-consuming Monte-Carlo method.

- This lecture begins to introduce those daunting notations and mathematical treatment for reinforcement learning.

Reward

- Let r_t be the spending on day-t. That is, r_t is *cost* or *reward* for time t.
- The reward r_t is fully determined by the state at time t, by a function $R(\cdot)$ such as $r_t = R(s)$.

Definition 1 (reward function)

A real-valued function $R : S \to \mathbb{R}$ is called a *reward function* that determines the reward given the state. That is, $R(s) = \mathbb{E}[r_t | S_t = s]$

Markov reward process (MRP)

Definition 2 (Markov reward process (MRP))

A MRP refers to a reward process where the underlying stochastic process is characterized with Markov property.

Remark 1

In other words, MRP is a reward process where the reward is determined by DTMC's state.

Return

- We were asked to find the expected value of $r_0 + r_1 + \cdots + r_9$.

Definition 3 (return)

The *return* G_t is the sum of remaining reward at time t.

- Using this notation, our problem has following returns.
  - $G_0 = r_0 + r_1 + \cdots + r_9$
  - $G_1 = r_1 + \cdots + r_9$
  - $G_2 = r_2 + \cdots + r_9$
  - $\cdots$
  - $G_9 = r_9$
- In other words, we were asked the value of $\mathbb{E}[G_0 | S_0 = c]$.

Dependence

- In our problem, we were asked to find the expected value of G_0 starting from state c at time 0.
- At time 0, the value of r_0 is known, but $r_1, \ldots, r_9$ are random variables. So, G_0 is random variable as well.

- The random variable G_0 depends on
  - the current state S_0
  - and the randomness along the stochastic path.
- In general, the random variable G_t depends on
  - the last-known state S_t
  - and some randomness along the remaining path.

- Since G_t is a random variable, we want to evaluate $\mathbb{E}[G_t]$.
- In general, considering its dependence structure, we are interested in evaluating $\mathbb{E}[G_t | S_t = s]$.

State-value function

- The current problem is $\mathbb{E}[r_0 + r_1 + \cdots + r_9 | S_0 = c]$ or $\mathbb{E}[G_0 | S_0 = c]$.
- This motivates the following definition.

Definition 4 (state-value function)

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

- Then, we are interested in finding $V_0(c) = \mathbb{E}[G_0 | S_0 = c] = \mathbb{E}[r_0 + \cdots + r_9 | S_0 = c]$.

II. Method 1 - Monte-Carlo simulation

Recap

- The MC simulation was a valid approach. We shall review our initial effort with newly introduced terminology.
- The algorithm includes$\cdots$
  1. Generate a single stochastic path starting from the *initial state*, $S_0 = c$.
  2. Collect a single value of *return*, $G^{i-th}, 1 \leq i \leq$ `MC_N`, by accumulating *rewards*, $\{r_0, r_1, ..., r_9\}$, along the path.
  3. Take an average of collected *returns* to evaluate state-value function, $V_0(c)$.

```r
MC_N <- 10000
spending_records <- rep(0, MC_N)
for (i in 1:MC_N) {
  path <- "c" # coke today (day-0)
  for (t in 1:9) {
    this_state <- str_sub(path, -1, -1)
    next_state <- soda_simul(this_state)
    path <- paste0(path, next_state)
  }
  spending_records[i] <- cost_eval(path)
}

cost_eval <- function(path) {
  cost_one_path <-
    str_count(path, pattern = "c")*1.5 +
    str_count(path, pattern = "p")*1
  return(cost_one_path)
}

mean(spending_records)
```

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.

Exercise 1

Write a python code for previous page's Pseudo code. Try to use the same variable names.

III. Method 2 - Iterative solution

Math review

Conditional expectation

- For a partition E_1 and E_2, ($E_1 \cap E_2 = \emptyset$ and $E_1 \cup E_2 = U$)
  - $\mathbb{P}(A) = \mathbb{P}(A|E_1)\mathbb{P}(E_1) + \mathbb{P}(A|E_2)\mathbb{P}(E_2)$
  - $\mathbb{E}(X) = \mathbb{E}(X|E_1)\mathbb{P}(E_1) + \mathbb{E}(X|E_2)\mathbb{P}(E_2)$
  - $\mathbb{E}(X|A) = \mathbb{E}(X|E_1 \cap A)\mathbb{P}(E_1|A) + \mathbb{E}(X|E_2 \cap A)\mathbb{P}(E_2|A)$

Conditional expectation is linear as well.

- $\mathbb{E}(X + Y) = \mathbb{E}X + \mathbb{E}Y$
- $\mathbb{E}(X + Y|A) = \mathbb{E}[X|A] + \mathbb{E}[Y|A]$

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,
        - $V_9(c) = \mathbb{E}[r_9|S_9 = c] = R(c) = 1.0$ and
        - $V_9(p) = \mathbb{E}[r_9|S_9 = p] = R(p) = 1.5$.
    - 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 $\mathbb{E}(X|A) = \mathbb{E}(X|E_1, A)\mathbb{P}(E_1|A) + \mathbb{E}(X|E_2, A)\mathbb{P}(E_2|A)$)

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

- (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',

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

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

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

- 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).

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

- For general t, *(exercise)*

- So far,

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

- 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

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

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

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

simply a system of linear equations!

```r
P <- array(c(0.7,0.5,0.3,0.5), dim=c(2,2))
R <- array(c(1.5,1.0), dim=c(2,1))
H <- 10 # time-horizon
v_t1 <- array(c(0,0), dim=c(2,1)) # v_{t+1}

t <- H-1
while (t >= 0) {
  v_t  <- R + P %*% v_t1
  t    <- t-1
  v_t1 <- v_t
}
v_t
```

```
##          [,1]
## [1,] 13.35937
## [2,] 12.73438
```

- Thus, we have the following *state-value function*.
  - $V_0(c) = 13.359375$
  - $V_0(p) = 12.734375$

Backward induction for estimating *state-value function*

- Formally, for a *finite-horizon MRP*, the following is *backward induction* for estimating *state-value function*.

```
# Backward induction for state-value function
# with transition prob mat P, reward vector R, time-horizon H, state-value vector v_{}
v_H <- zero-column vector
t <- H-1
while t >= 0
  v_t <- R + P*v_{t+1}
  t <- t-1
return v_t # this is v_0(s) for all s, because t=0 at this point
```

Summary and Discussion

- In this lecture, we dealt with the following question.

  Given I drink coke today, what is likely my consumption for upcoming 10 days? (Pepsi is $1 and Coke is $1.5)

- The first approach was Monte-Carlo simulation and The second approach was iterative solution methods.
- Both approaches were possible because the time horizon was finite. If the time horizon was infinite, then Monte-Carlo approach can't really complete a stochastic path. If the time horizon was infinite, then the iterative method cannot find the terminal period, which serves as a initial point of iteration.
- However, the infinite horizon problem is easier to solve. Next lecture will define infinite horizon problem and discusses the approach to evaluate the state-value function.

```
"Success isn't permarnent, and failure isn't fatal. - Mike Ditka"
```