D1_DTMC Python

Jaemin Park

2021-01-13

차 례

Exercises

p.10 MC Simulator

R code

```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)
}
```

Python code

```python
def cost_eval(path):
    cost_one_path = path.count("c")*1.5 + path.count("p")*1
    return cost_one_path
MC_N = 100
record = np.array([])
for i in range(MC_N):
    this_stage = "c"
    result = []
    for j in range(9):
        result.append(this_stage)
        next_state = soda_simul(this_stage)
        this_stage = next_state
    record = np.append(record, np.array([cost_eval(''.join(result))]))
```

p.11 MC Simulator

R code

```
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)
```

Python code

```
#MC evalutaion for state-value function
#with state s, time 0, reward r, time horizon H
episode = 100
episode_i = 0
cum_sum_G_i = 0
while episode_i < episode:
  this_stage = "c"
  result = []
  for j in range(9):
      result.append(this_stage)
      next_state = soda_simul(this_stage)
      this_stage = next_state

  G_i = cost_eval(result)
  cum_sum_G_i = cum_sum_G_i + G_i
  episode_i +=1
V_t = cum_sum_G_i / episode
V_t
```

```
## 12.045
```

p.18 Exercise

For general t,(exercise)

$$\begin{aligned}
V_t(S) &= \mathbb{E}[G_t \mid S_t = s] \\
&= \mathbb{E}[\sum_{i=t}^{9} r_t \mid S_t = s] \\
&= \mathbb{E}[r_t + r_{t+1} + .. + r_9 \mid S_t = s] \\
&= \mathbb{E}[r_t \mid S_t = s] + \mathbb{E}[r_{t+1} + .. + r_9 \mid S_t] \\
&= R(s) + \sum_{s' \in S} P_{ss'}\, \mathbb{E}[G_{t+1} \mid S_t = s, S_{t+1} = s'] \\
&= R(s) + \sum_{s' \in S} P_{ss'}\, \mathbb{E}[G_{t+1} \mid S_{t+1} = s'](\because Markov\, property) \\
&= R(s) + \sum_{s' \in S} P_{ss'} V_{t+1}(s')
\end{aligned}$$

p.20 Iterative solution

```
import numpy as np

P = np.matrix([[0.7,0.3],[0.5,0.5]])
R = np.matrix([[1.5],[1]])
H=10
v_t1 = np.matrix([[0],[0]])

t=H-1
while(t>=0):
    v_t = R + P*v_t1
    t-=1
    v_t1 = v_t

print(v_t)
```

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