Lecture F3. MDP without Model - Policy Iteration (MC, TD)

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

SEOULTECH 서울과학기술대학교 데이터사이언스학과

- `skiier.R` is loaded as follows.

```
source("../skiier.R")
```

```
## [1] "Skiier's problem is set."
## [1] "Defined are `state`, `P_normal`, `P_speed`, `R_s_a`, `q_s_a_init` (F2, p15)."
## [1] "Defined are `pi_speed`, and `pi_50` (F2, p16)."
## [1] "Defined are `simul_path()` (F2, p17)."
## [1] "Defined are `simul_step()` (F2, p18)."
## [1] "Defined are `pol_eval_MC()` (F2, p19)."
## [1] "Defined are `pol_eval_TD()` (F2, p20)."
## [1] "Defined are `pol_imp()` (F2, p20)."
```

21

I. Policy Iteration 1 - MC Control

Process

- The MC policy iteration process is summarized as follows:
  0. Initialize $q(s,a)$
  1. Begin with a policy π
  2. Generate a sample path using the current π (`simul_path()`)
  3. Evaluate the current policy π and update $q(s,a)$ (`pol_eval_MC()`)

$$q(s,a) \leftarrow q(s,a) + \alpha(G_t - q(s,a)), \ \forall s,a$$

  4. Improve the policy into a ϵ-greedy policy using $q(s,a)$ (`pol_imp()`)

$$\pi(s,a) \leftarrow argmax_{a\in\mathcal{A}} q(s,a)$$

  5. Repeat 2-4 many times

- We will let `num_ep` to grow from 10^3, 10^4, and $5 \cdot 10^4$.

1000 10000 50000

MC control w/ num-ep=1000

```
num_ep <- 10^3
beg_time <- Sys.time()
q_s_a <- q_s_a_init                 # 0. initialize q(s,a)
pi <- pi_50                         # 1. begin w/ policy π
for (epi_i in 1:num_ep) {
  sample_path_i <- simul_path(pi, P_normal, P_speed, R_s_a)          # 2. gen sample path.
  q_s_a <- pol_eval_MC(sample_path_i, q_s_a, alpha = 1/epi_i)        # 3. evaluate policy π
  pi <- pol_imp(pi, q_s_a, epsilon = 1/epi_i)                        # 4. improve π into a ε-greedy policy.
}                                                     GLIE가 만족됨
end_time <- Sys.time()
print(end_time-beg_time)
```

```
## Time difference of 0.2078 secs
```

```
t(pi)
```

```
##    0 10 20 30 40 50 60 70
## n 0  1  0  1  1  0  1  1
## s 1  0  1  0  0  1  0  0
```

speed: 0, 20, 50.

```
t(q_s_a)
```

```
##          0      10      20      30      40      50       60 70
## n -5.358 -4.004 -3.859 -2.650 -1.646 -1.608 -0.9929  0
## s -5.081 -4.006 -3.340 -3.004 -1.654 -1.606 -1.0003  0
```

MC control w/ num-ep = 10^4

```
num_ep <- 10^4
beg_time <- Sys.time()
q_s_a <- q_s_a_init
pi <- pi_50
for (epi_i in 1:num_ep) {
  sample_path_i <- simul_path(pi, P_normal, P_speed, R_s_a)
  q_s_a <- pol_eval_MC(sample_path_i, q_s_a, alpha = 1/epi_i)
  pi <- pol_imp(pi, q_s_a, epsilon = 1/epi_i)
}
end_time <- Sys.time()
print(end_time-beg_time)
```

```
## Time difference of 1.811 secs
```

```
t(pi)
```

```
##   0 10 20 30 40 50 60 70
## n 0  0  1  1  1  0  1  1
## s 1  1  0  0  0  1  0  0
```

```
t(q_s_a)
```

```
##        0     10     20     30     40     50      60 70
## n -5.734 -4.500 -3.679 -2.677 -1.675 -1.678 -0.9206  0
## s -5.338 -2.969 -4.005 -2.726 -2.018 -1.670 -1.7399  0
```

GLIE $\frac{1}{i} \to 0$, $\sum_{i=1}^{\infty} \frac{1}{i} = \infty$

Current $(0.9997)^i \to 0$, $\sum (0.9997)^i < \infty$

```
num_ep <- 5*10^4
beg_time <- Sys.time()
q_s_a <- q_s_a_init
pi <- pi_50
exploration_rate <- 1
for (epi_i in 1:num_ep) {
  sample_path_i <- simul_path(pi, P_normal, P_speed, R_s_a)
  q_s_a <- pol_eval_MC(sample_path_i, q_s_a, alpha = 1/epi_i)
  pi <- pol_imp(pi, q_s_a, exploration_rate)
  exploration_rate <- max(exploration_rate*0.9997, 0.001) # exponential decay # 0.9997^10000=0.050
}
end_time <- Sys.time()
print(end_time-beg_time)
```

$1 \to 0.9997 \to 0.9997^2 \to$

```
## Time difference of 8.18 secs
```

```
t(pi)
```

```
##   0 10 20 30 40 50 60 70
## n 0  1  0  1  0  0  1  1
## s 1  0  1  0  1  1  0  0
```

```
t(q_s_a)
```

```
##         0     10     20     30    40     50      60 70
## n -6.044 -4.768 -4.176 -2.781 -1.92 -2.131 -0.9983  0
## s -5.146 -5.795 -3.474 -3.568 -1.69 -1.710 -1.5800  0
```

II. Policy Iteration 2 - TD Control (a.k.a. sarsa)

Process

- The TD policy iteration process is summarized as follows:
  0. Initialize $q(s, a)$
  1. Begin with a policy π
  2. Begin a new sample path from the state s
  3. Proceed a time step to generate subsequent a, r, s', a' (`simul_step()`)
  4. Evaluate the current policy π and update $q(s, a)$ (`pol_eval_TD()`)

$$q(s, a) \leftarrow q(s, a) + \alpha(r_t + \gamma q(s', a') - q(s, a)), \ \forall s, a$$

  5. Improve the policy into a ϵ-greedy policy using $q(s, a)$ (`pol_imp()`)
  6. Repeat 3-5 until the episode ends.
  7. Repeat 2-6 many times (*why not until policy converges?*)
- Likewise, We will let `num_ep` to grow from 10^3, 10^4, and $5 \cdot 10^4$.

```
num_ep <- 10^3
beg_time <- Sys.time()
q_s_a <- q_s_a_init      #0
pi <- pi_50              #1.
for (epi_i in 1:num_ep) {
  s_now <- "0"           #2
  while (s_now != "70") {
    sample_step <- simul_step(pi, s_now, P_normal, P_speed, R_s_a)  #3
    q_s_a <- pol_eval_TD(sample_step, q_s_a, alpha = 1/epi_i)       #4.
    pi <- pol_imp(pi, q_s_a, epsilon = 1/epi_i)                     #5.
    s_now <- sample_step[4]  v
  }
}
end_time <- Sys.time()
print(end_time-beg_time)
```

```
## Time difference of 0.2954 secs
```

```
t(pi)
```

```
##    0 10 20 30 40 50 60 70
## n 0  1  0  1  1  0  1  1
## s 1  0  1  0  0  1  0  0
```

```
t(q_s_a)
```

```
##         0     10     20     30     40     50      60 70
## n -3.511 -2.946 -2.533 -2.028 -1.335 -1.536 -0.9711  0
## s -3.511 -2.947 -2.533 -2.028 -1.335 -1.535 -1.5441  0
```

```r
num_ep <- 10^4
beg_time <- Sys.time()
q_s_a <- q_s_a_init
pi <- pi_50
for (epi_i in 1:num_ep) {
  s_now <- "0"
  while (s_now != "70") {
    sample_step <- simul_step(pi, s_now, P_normal, P_speed, R_s_a)
    q_s_a <- pol_eval_TD(sample_step, q_s_a, alpha = 1/epi_i)
    pi <- pol_imp(pi, q_s_a, epsilon = 1/epi_i)
    s_now <- sample_step[4]
  }
}
end_time <- Sys.time()
print(end_time-beg_time)
```

```
## Time difference of 2.655 secs
```

```r
t(pi)
```

```
##   0 10 20 30 40 50 60 70
## n 0  1  1  0  1  0  1  1
## s 1  0  0  1  0  1  0  0
```

```r
t(q_s_a)
```

```
##        0     10     20     30     40     50      60 70
## n -3.943 -3.359 -2.937 -2.412 -1.566 -1.631 -0.9951  0
## s -3.943 -3.359 -2.937 -2.412 -1.566 -1.630 -0.9952  0
```

```
num_ep <- 10^5
beg_time <- Sys.time()
q_s_a <- q_s_a_init
pi <- pi_50
exploration_rate <- 1
for (epi_i in 1:num_ep) {
  s_now <- "0"
  while (s_now != "70") {
    sample_step <- simul_step(pi, s_now, P_normal, P_speed, R_s_a)
    q_s_a <- pol_eval_TD(sample_step, q_s_a, alpha = max(1/epi_i, 0.01))
    pi <- pol_imp(pi, q_s_a, epsilon = exploration_rate)
    s_now <- sample_step[4]
    exploration_rate <- exploration_rate*0.9995
  }
}
end_time <- Sys.time()
print(end_time-beg_time)
```

```
## Time difference of 21.93 secs
```

```
t(pi)
```

```
##   0 10 20 30 40 50 60 70
## n 0  1  0  1  1  0  1  1
## s 1  0  1  0  0  1  0  0
```

```
t(q_s_a)
```

```
##          0     10     20     30     40     50      60 70
## n -5.382 -4.478 -3.687 -2.692 -1.653 -1.887 -0.9996  0
## s -5.054 -4.561 -3.401 -2.801 -1.769 -1.660 -1.4229  0
```

Exercise 1

Write the python code for this lecture note (both MC and TD control)

Exercise 2

The previous Part E provides us the correct solution. In your python code for MC and TD control, you have freedom to modify 1) number of iteration and 2) the exploration decaying scheme. Modify the two and see if you can match the results we obtained in Part E.

```
"It's not that I'm so smart, it's just that I stay with problems longer. - A. Einstein"
```