Lecture F3. MDP without Model 3

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)."
```

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()`)

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

  5. Repeat 2-4 many times

```
num_ep <- 10^3
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 0.9634 secs
```

```
t(pi)
```

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

```
t(q_s_a)
```

```
##        0     10     20     30     40     50      60 70
## n -5.581 -4.456 -5.351 -4.947 -3.011 -3.998 -0.9788  0
## s -7.493 -4.462 -3.467 -3.019 -1.684 -1.593 -2.9792  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) {
  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 8.288 secs
```

```r
t(pi)
```

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

```r
t(q_s_a)
```

```
##        0     10     20     30     40     50      60 70
## n -5.449 -4.409 -4.201 -2.667 -1.731 -1.698 -0.9996  0
## s -5.907 -4.409 -3.423 -3.342 -1.669 -1.900 -1.0541  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) {
  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 <- exploration_rate*0.9995 # exponential decay
}
end_time <- Sys.time()
print(end_time-beg_time)
```

```
## Time difference of 1.165 mins
```

```
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 -5.721 -4.197 -4.005 -2.755 -1.892 -2.264 -1.000  0
## s -5.117 -4.197 -3.449 -2.976 -1.670 -1.466 -1.581  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?)*

```r
num_ep <- 10^3
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 0.765 secs
```

```r
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
```

```r
t(q_s_a)
```

```
##        0     10     20     30     40     50      60 70
## n -3.296 -2.792 -2.514 -2.142 -1.441 -1.639 -0.9875  0
## s -3.295 -2.792 -2.513 -2.143 -1.441 -1.635 -1.5000  0
```

```
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 6.834 secs
```

```
t(pi)
```

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

```
t(q_s_a)
```

```
##        0   10     20     30     40     50      60 70
## n -3.909 -3.3 -2.868 -2.347 -1.535 -1.643 -0.9932  0
## s -3.909 -3.3 -2.868 -2.347 -1.535 -1.643 -0.9933  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 58.54 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 -5.352 -4.493 -3.682 -2.695 -1.683 -1.924 -0.9999  0
## s -5.329 -4.535 -3.540 -2.801 -1.761 -1.669 -1.5680  0
```

Exercise 1

Feel free to try different schemes for the number of iterations and exploration decaying scenarios.

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