Lecture H1. Value-based agent 1

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

SEOULTECH 서울과학기술대학교 데이터사이언스학과

- `skiier.R` is loaded as follows.

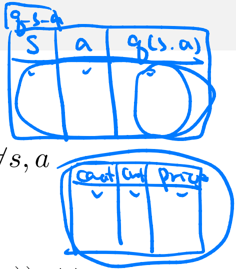
```
source("../skiier.R")
```

```
## [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. Algorithms overview

II. Deep Q-Learning

Motivation



- (`pol_eval_MC()`) MC control updates $q(s, a)$:

$$q(s,a) \leftarrow q(s,a) + \alpha(\underbrace{G_t}_{\text{mc-tgt}} - q(s,a)), \ \forall s, a$$

- (`pol_eval_TD()`) TD control updates $q(s, a)$:

$$q(s,a) \leftarrow q(s,a) + \alpha(\underbrace{r_t + \gamma q(s', a')}_{\text{TD-tgt}} - q(s,a)), \ \forall s, a$$

- (`pol_eval_Q()`) Q-learning updates $q(s, a)$:

$$q(s,a) \leftarrow q(s,a) + \alpha(\underbrace{r_t + \gamma max_{a' \in \mathcal{A}} q(s', a')}_{\text{Q-tgt}} - q(s,a)), \ \forall s, a$$

- Deep Q-learning
  - It approximates the $q(\cdot, \cdot)$ function with deep neural network.
  - In other words, $q(\cdot, \cdot)$ approximates 'Q-target', $r_t + \gamma max_{a' \in \mathcal{A}} q(s', a')$, using her experience.
  - Improve her policy using the $q(\cdot, \cdot)$ network.

Recap

- pol_eval_Q() (F4, p6) ✓

```
pol_eval_Q <- function(sample_step, q_s_a, alpha) {
  s <- sample_step[1]
  a <- sample_step[2]
  r <- sample_step[3] %>% as.numeric()
  s_next <- sample_step[4]
  q_s_a[s,a] <- q_s_a[s,a] + alpha*(r+max(q_s_a[s_next,])-q_s_a[s,a])
  return(q_s_a)
}
```

- pol_imp() (F2, p21) ✓

```
pol_imp <- function(pi, q_s_a, epsilon) { # epsilon = exploration_rate
  for (i in 1:nrow(pi)) {
    if (runif(1) > epsilon) { # exploitation
      pi[i, which.max(q_s_a[i,])] <- 1
      pi[i,-which.max(q_s_a[i,])] <- 0
    } else { # exploration
      pi[i,] <- 1/ncol(pi)
    }
  }
  return(pi)
}
```

Recap

- Q-learning (F4, p7)

```
num_ep <- 10^5
beg_time <- Sys.time()
q_s_a <- q_s_a_init #1.
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_Q(
      sample_step, q_s_a, alpha = max(1/epi_i, 0.05)) #2.
    if (epi_i %% 100 == 0) {
      pi <- pol_imp(pi, q_s_a, epsilon = exploration_rate) #3.
    }
    s_now <- sample_step[4]
    exploration_rate <- max(exploration_rate*0.9995, 0.001)
  }
}
```

- Strategy
  1. `q_s_a` needs to be defined as a DQN.
  2. `pol_eval_DQN()` needs to be written.
  3. `pol_imp_DQN()` needs to be written.

0. Prep

- `vec2mat()` is to prepare input for `q_net`

```
vec2mat <- function(vec) {
  return(matrix(vec, nrow=1, ncol=length(vec)))
}
a <- vec2mat(rep(0,8))
class(a)
```

```
## [1] "matrix" "array"
```

```
dim(a)
```

```
## [1] 1 8
```

1. q_s_a needs to be defined as a DQN.

- Recap: q_s_a for $q(s, a)$

```
q_s_a_init <- cbind(rep(0,length(states)), rep(0,length(states)))
rownames(q_s_a_init) <- states
colnames(q_s_a_init) <- c("n", "s")
t(q_s_a_init)
```

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

- How to shape this into input-output structure?

```
library(data.table)
library(mltools)
action <- factor(c("n", "s"))
q_net_sketch <- data.frame(s = factor(states), a = factor(action)) %>%
  as.data.table()
q_net_sketch %>% one_hot() %>% colnames()
```

```
##  [1] "s_0"  "s_10" "s_20" "s_30" "s_40" "s_50" "s_60" "s_70" "a_n"  "a_s"
```

- q_net() for q_s_a

```
library(keras)
q_net <- keras_model_sequential() %>%
  layer_dense(units = 16, input_shape = c(8), activation = "relu") %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(1, activation = "linear")
q_net %>% compile(loss = "mse", optimizer = "adam")
summary(q_net)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                     Output Shape                     Param #
## ================================================================================
## dense (Dense)                    (None, 16)                       144
## ________________________________________________________________________________
## dense_1 (Dense)                  (None, 16)                       272
## ________________________________________________________________________________
## dense_2 (Dense)                  (None, 1)                        17
## ================================================================================
## Total params: 433
## Trainable params: 433
## Non-trainable params: 0
## ________________________________________________________________________________
```

2. `pol_eval_DQN()` needs to be written.

- `pol_eval_Q()`

```
pol_eval_Q <- function(sample_step, q_s_a, alpha)
  s <- sample_step[1]
  a <- sample_step[2]
  r <- sample_step[3] %>% as.numeric()
  s_next <- sample_step[4]
  q_s_a[s,a] <- q_s_a[s,a] +
    alpha*(r+max(q_s_a[s_next,])-q_s_a[s,a])
  return(q_s_a)
}
```

- `pol_eval_DQN()`

```
pol_eval_DQN <- function(sample_step, q_net) {
  s <- sample_step[1] %>% as.numeric()
  a <- sample_step[2]
  r <- sample_step[3] %>% as.numeric()
  s_next <- sample_step[4] %>% as.numeric()
  # Prepare `s_a_one_hot`
  s_a_one_hot <- c(rep(0,8))
  if (s < 70) s_a_one_hot[s/10+1] <- 1
  if (a=="n") s_a_one_hot[8] <- 1
  # Calculate `Q_tgt`
  s_next_one_hot <- c(rep(0,7))
  if (s_next < 70) {
    s_next_one_hot[s_next/10+1] <- 1
  }
  Q_tgt <- r + max(
    q_net %>% predict(vec2mat(c(s_next_one_hot,1))),
    q_net %>% predict(vec2mat(c(s_next_one_hot,0))))
  # Update `q_net`
  q_net %>% fit(vec2mat(s_a_one_hot), Q_tgt,
                epoch=1, verbose = 0)
}
```

3. pol_imp_DQN() needs to be written.

- pol_imp()

```
pol_imp <- function(pi, q_s_a, epsilon) {
  for (i in 1:nrow(pi)) {
    if (runif(1) > epsilon) { # exploitation
      pi[i, which.max(q_s_a[i,])] <- 1
      pi[i,-which.max(q_s_a[i,])] <- 0
    } else { # exploration
      pi[i,] <- 1/ncol(pi)
    }
  }
  return(pi)
}
```

- pol_imp_DQN()

```
pol_imp_DQN <- function(pi, q_net, epsilon) {
  for (i in 1:nrow(pi)) {
    if (runif(1) > epsilon) { # exploitation
      s <- rownames(pi)[i] %>% as.numeric()
      s_one_hot <- c(rep(0,7))
      if (s < 70) s_one_hot[s/10+1] <- 1
      idx <- which.max(
        c(q_net %>% predict(
            vec2mat(c(s_one_hot, 1))),
          q_net %>% predict(
            vec2mat(c(s_one_hot, 0)))))
      pi[i,  idx] <- 1
      pi[i, -idx] <- 0
    } else { # exploration
      pi[i,] <- 1/ncol(pi)
    }
  }
  return(pi)
}
```

Deep Q-Learnig

```r
num_ep <- 500
q_net <- keras_model_sequential() %>% #1.
  layer_dense(16, input_shape = c(8),
              activation = "relu") %>%
  layer_dense(16, activation = "relu") %>%
  layer_dense(1, activation = "linear")
q_net %>%
  compile(loss = "mse", optimizer = "adam")
pi <- pi_50
explore_rate <- 1
```

```r
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)
    pol_eval_DQN(sample_step, q_net) #2.
    s_now <- sample_step[4]
  }
  q_net %>% fit(
    vec2mat(c(rep(0,7),1)),0, epoch=1, verbose=0)
  q_net %>% fit(
    vec2mat(c(rep(0,7),0)),0, epoch=1, verbose=0)
  if (epi_i %% 10 == 0) { #3.
    pi <- pol_imp_DQN(
      pi, q_net, epsilon = explore_rate)
  }
  explore_rate <- max(explore_rate*0.9995, 0.001)
}
```

Results

```
q_s_a_DQN <- pi
for (i in 1:nrow(q_s_a_DQN)) {
  s <- rownames(q_s_a_DQN)[i] %>% as.numeric()
  s_one_hot <- c(rep(0,7))
  if (s < 70) s_one_hot[s/10+1] <- 1
  q_s_a_DQN[i,1] <- q_net %>% predict(vec2mat(c(s_one_hot,1)))
  q_s_a_DQN[i,2] <- q_net %>% predict(vec2mat(c(s_one_hot,0)))
}
q_s_a_DQN
```

```
##          n        s
## 0  -5.79983 -5.35285
## 10 -4.78158 -4.92658
## 20 -3.90191 -3.61255
## 30 -2.97708 -3.50893
## 40 -1.78193 -1.90130
## 50 -1.86362 -1.72882
## 60 -0.91133 -1.78539
## 70  0.04338  0.03107
```

$\boxed{-1.67}$

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