Tutorial 2.3 - Upper Confidence Bound

Tayyip Altan

October 2024

In this tutorial, we will be covering the Upper Confidence Bound (UCB) method to solve multi-armed bandit problems. We will be using the contextual package again. If you had trouble installing it, please visit the discussion board on Canvas to see additional hints.

Dataset

For this tutorial, we will work with the Yahoo dataset, which was also used in tutorial 2.1.

```r
# reads in full csv of Yahoo dataset
dfYahoo <- read.csv('yahoo_day1_10arms_tiny.csv')[,-c(1,2)]

# selects the two relevant columns from Yahoo dataset; arm shown to user and reward observed
dfYahoo_for_sim <- dfYahoo %>% select(arm, reward)
dfYahoo_for_sim$index <- 1:nrow(dfYahoo_for_sim)
```

UCB: notation

As covered in the lecture, UCB methods decide on the arm to pick using an 'optimistic' estimate of the reward - e.g. which arm has the most potential to yield a high reward? This is done by picking the arm that has the highest combination of (1) the estimated reward, $Q_t(a)$, and (2) a bonus that rewards the arm for being underexplored, denoted as $U_t(a)$. The UCB algorithm picks the arm that maximizes the sum of these two factors:

$$a_t = \text{argmax}_{a \in \mathcal{A}} Q_t(a) + c \cdot U_t(a). \tag{1}$$
$$\tag{2}$$

In the general case of the UCB algorithm, $U_t(a) = \sqrt{\frac{log(t)}{N_t(a)}}$, where $N_t(a)$ is the number of times arm a has been pulled so far.

Task 1: select the arm according to the UCB algorithm based on the data below. Here, the code already provides a dataframe with per arm (1) the average reward, (2) the number of pulls thus far, and (3) an given value for t. Use this dataframe to determine which arm should be pulled if $c = 0.1$. The selected arm should be 8.

```r
# set the seed
set.seed(0)

# get per item, the average reward and the number of items observed
dfYahoo_summary <- dfYahoo %>%
  group_by(arm) %>%
  summarise(avg_reward = mean(reward),
            n_pulls= n())

# the t in this case is simply the total of observations
t <- sum(dfYahoo_summary$n_pulls)
c <- 0.1

# TODO: Select an arm based on the UCB criterion (using equation (1))
```

UCB: code

We will be simulating the performance of the UCB algorithm on the Yahoo! data, again with the contextual package.

```r
# set the seed
set.seed(0)

## OfflineReplayEvaluatorBandit: simulates a bandit based on provided data
#
# Arguments:
#
#   data: dataframe that contains variables with (1) the reward and (2) the arms
#
#   formula: should consist of variable names in the dataframe. General structure is:
#       reward variable name ~ arm variable name
#
#   randomize: whether or not the bandit should receive the data in random order,
#              or as ordered in the dataframe.
#

# in our case, create a bandit for the data with 10 arms,
#   formula = reward ~ arm,
#   no randomization
bandit_yahoo <- OfflineReplayEvaluatorBandit$new(formula = reward ~ arm,
                                                 data = dfYahoo,
                                                 randomize = FALSE
                                                 )


# lets generate a 10 simulations, each of size 10.000,
size_sim <- 10000
n_sim <- 10

# here we define the UCB policy object
# for some reason, the contextual package uses the 'UCB2Policy' tag for standard UCB algorithm
# in the package, 'alpha' is what we refer to as c
```

```r
UCB_01          <- UCB2Policy$new(alpha=0.1)

# the contextual package works with 'agent' objects - which consist of a policy and a bandit
UCB_01_agent <- Agent$new(UCB_01, # add our UCB1 policy
                        bandit_yahoo) # add our bandit

## simulator: simulates a bandit + policy based on the provided data and parameters
#
# Arguments:
#
#   agent: the agent object, previously defined
#
#   horizon: how many observations from dataset used in the simulation
#
#   do_parallel: if True, runs simulation in parallel
#
#   simulations: how many simulations?
#

simulator          <- Simulator$new(UCB_01_agent, # set our agent
                                      horizon= size_sim, # set the sizeof each simulation
                                      do_parallel = TRUE, # run in parallel for speed
                                      simulations = n_sim # simulate it n_sim times
                                  )


# run the simulator object
history_UCB_01            <- simulator$run()

# gather results
df_Yahoo_UCB_01 <- history_UCB_01$data %>%
  select(t, sim, choice, reward, agent)
```

In the dataframe *df_Yahoo_UCB_01* we have gathered the results of the UCB policy for $c = 0.1$.

Task 2: repeat the steps of this tutorial, but now for $c = 0.5$.

```r
# set the seed
set.seed(0)

# TODO: Adjust the code in this chunk to fit to the UCB policy for c = 0.5

# Define the policy
UCB_05          <- UCB2Policy$new(alpha=0.1)

# Define the agent
UCB_05_agent <- Agent$new(UCB_01, bandit_yahoo)

# Simulate
simulator          <- Simulator$new(list(UCB_01_agent, UCB_05_agent), # set our agents
                                    horizon= size_sim, # set the sizeof each simulation
                                    do_parallel = TRUE, # run in parallel for speed
                                    simulations = n_sim, # simulate it n_sim times
                                  )
```

```r
# run the simulator object
history_UCB           <- simulator$run()

# gather results
df_Yahoo_UCB <- history_UCB$data %>%
  select(t, sim, choice, reward, agent)
```

Task 3: Make a plot that compares the UCB policy for $c = 0.1$, $c = 0.5$. Compare the policies based on average cumulative reward. Can you conclude which one performs better, and if so why?

Your answer to Task 3 here:

```r
# agent replaced with alpha value
df_Yahoo_UCB$agent <- recode(df_Yahoo_UCB$agent, 'UCB2' = '0.1', 'UCB2.2'='0.5')

# Maximum number of observations
max_obs <- 700

#  data.frame aggregated for two agents
df_history_agg <- df_Yahoo_UCB %>%
  group_by(agent, sim)%>% # group by number of arms, the sim
  mutate(cumulative_reward = cumsum(reward))%>% # calculate cumulative sum
  group_by(agent, t) %>% # group by number of arms, the t
  summarise(avg_cumulative_reward = mean(cumulative_reward),# calc cumulative reward, se, CI
            se_cumulative_reward = sd(cumulative_reward, na.rm=TRUE)/sqrt(n_sim)) %>%
  mutate(cumulative_reward_lower_CI =avg_cumulative_reward - 1.96*se_cumulative_reward,
         cumulative_reward_upper_CI =avg_cumulative_reward + 1.96*se_cumulative_reward)%>%
  filter(t <=max_obs)

# TODO: make a plot using ggplot to compare UCB policy for c=0.1 and c=0.5
# 1: A plot that shows only the average cumulative rewards over time using the df_history_agg dataframe
# 2: The plot as defined in (1) together with the 95\% confidence interval.
```

Task 4: compare the UCB policy for $c = 0.1$ to an ϵ-greedy policy where $\epsilon = 0.1$, using the contextual package. Which one does better? Why do you think this might be?. Repeat this exercise again with *horizon* of 10.000.

```r
# set the seed
set.seed(0)

# TODO: Add the EpsilonGreedyPolicy below to define the policy
eps_greedy <-

# TODO: Add the policy and the bandit below to define the agent
eps_greedy_agent <- Agent$new(, # Add the policy
                              ) # Add the bandit

# TODO: Finish the Simulator below
simulator <- Simulator$new(list(UCB_01_agent, eps_greedy_agent), # set our agents
                           horizon= , # set the sizeof each simulation
                           do_parallel = , # run in parallel for speed
                           simulations = , # simulate it n_sim times
                           )

# TODO: run the simulator object
history_compare_UCB_eps_greedy <-

# TODO: select relevant columns of the simulation
df_Yahoo_UCB_eps_greedy <-
```

```r
# data.frame aggregated for two agents: UCB and epsilon greedy
df_history_agg <- df_Yahoo_UCB_eps_greedy %>%
  group_by(agent, sim)%>% # group by number of arms, the sim
  mutate(cumulative_reward = cumsum(reward))%>% # calculate cumulative sum
  group_by(agent, t) %>% # group by number of arms, the t
  summarise(avg_cumulative_reward = mean(cumulative_reward),# calc cumulative reward, se, CI
            se_cumulative_reward = sd(cumulative_reward, na.rm=TRUE)/sqrt(n_sim)) %>%
  mutate(cumulative_reward_lower_CI =avg_cumulative_reward - 1.96*se_cumulative_reward,
         cumulative_reward_upper_CI =avg_cumulative_reward + 1.96*se_cumulative_reward)%>%
  filter(t <=max_obs)

# TODO: Make a plot to compare the performance of the UCB policy and epsilon-greedy policy
# 1: A plot that shows only the average cumulative rewards over time using the df_history_agg dataframe
# 2: The plot as defined in (1) together with the 95\% confidence interval.
```

Task 5: to better understand the ϵ-greedy policy and UCB algorithm, plot for each the cumulative % of the time a certain arm is chosen. Which arm is chosen most often? does this differ between the two approaches?

```r
# dataframe with arm choices for UCB
df_arm_choices_UCB_01 <- df_Yahoo_UCB_01%>%
  group_by(t, sim,choice, .drop=FALSE) %>%
  mutate(n = sum(!is.na(reward))) %>%   # Count valid rewards
  group_by(t, choice, .drop=FALSE)%>%
  summarise(sum_n = sum(n)) %>%   # Sum the counts of chosen arms
  group_by(choice, .drop=FALSE)%>%
  mutate(cum_sum_n = cumsum(sum_n), # Cumulative sum of chosen arms
         avg_cum_sum_n = cum_sum_n / (10 *t)) %>% # Average cumulative selections
  filter(t <=max_obs)             # Filter by max observations

# TODO: Plot the average % of arms chosen for the UCB policy
```

```r
# dataframe with arm choices for Epsilon Greedy
df_arm_choices_EPS_01 <- df_Yahoo_UCB_eps_greedy %>%
  group_by(t, sim,choice, .drop=FALSE) %>%
  mutate(n = sum(!is.na(reward))) %>%   # Count valid rewards
  group_by(t, choice, .drop=FALSE)%>%
  summarise(sum_n = sum(n)) %>%   # Sum the counts of chosen arms
  group_by(choice, .drop=FALSE)%>%
  mutate(cum_sum_n = cumsum(sum_n), # Cumulative sum of chosen arms
         avg_cum_sum_n = cum_sum_n / (10 *t)) %>% # Average cumulative selections
  filter(t <=max_obs)             # Filter by max observations

# TODO: Plot the average % of arms chosen for the Epsilon Greedy policy
```