Tutorial 2.2 - Thompson Sampling

Tayyip Altan

October 2024

In this tutorial, we will be covering the Thompson Sampling (TS) method to solve multi-armed bandit problems. Throughout this tutorial we will be using the *contextual package* in R to help us with our analysis. To install the package, download the latest version here: https://cran.r-project.org/src/contrib/Archive/contextual/?C=M;O=D

In this tutorial, we will be covering the Thompson Sampling (TS) method to solve multi-armed bandit problems.

Dataset

For this tutorial, we will work with a dataset from ZOZO, the largest Japanese fashion e-commerce company. The company uses multi-armed bandit algorithms to recommend fashion items to users on its large-scale e-commerce platform.

We will work with a version of the dataset that contains 10 fashion items. Each row contains a fashion item randomly shown to a user, and if the user clicked on said fashion item. The original dataset contains +170k observations. However, in this tutorial we will be using the version with 1k observations. The fashion items were randomly shown to users, which makes this dataset suitable for evaluating our TS algorithm.

```r
# this version contains 10 arms (ZOZO)
dfZozo_10 <- read.csv('zozo_noContext_10items_tiny.csv') %>%
  select(item_id, click)
```

Below the notation and code for Thompson Sampling is displayed.

Thompson Sampling: notation

When working with the ϵ-greedy algorithm, we estimate the expected reward using the average reward up until that point. With Thompson Sampling, we take a different approach; instead of focusing solely on the expected reward, we try to find out the distribution of the reward. This is done in three steps:

1. The reward $Q_t(a)$ for each arm follows a beta distribution. At the start, at $t = 0$, every arm is assumed to follow the same beta distribution with parameters $\alpha_0 = 1$ and $\beta_0 = 1$. This means we initially assume each arm has an equal probability of giving a reward.
2. At each time t, the algorithm samples n_{sample} observations from each of the distributions. The arm which has the highest average reward according to the sample is the chosen arm.
3. We then observe the reward r_t, and update the parameters of the distribution accordingly.

- $\alpha_t = \alpha_{t-1} + r_t$
- $\beta_t = \beta_{t-1} + 1 - r_t$

Task 1: select from the dataframe below an arm according to the Thompson Sampling algorithm. Per arm, the alpha and beta parameters have been given. Use $n_{\text{sample}} = 100$.

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

# arm index
arm <- 1:10

# alpha per arm
alpha <- c(10,2,9,3,1,8,7,4,6,5)

# beta per arm
beta <- c(1,8,3, 2,4,7,6,5,9,10)

# dataframe with alpha, beta per arm
df_alpha_beta <- data.frame(arm=arm,alpha = alpha, beta = beta)

# number of samples to draw
n_sample <- 100

# create dataframe with sampled values
df_sampled <- mapply(rbeta, n_sample, df_alpha_beta$alpha, df_alpha_beta$beta)

# TODO: select an arm based on step 2 of TS (the arm with the highest average reward)
```

Thompson Sampling: code

We will be simulating the performance of the Thompson Sampling algorithm on the Zozo data.

To do so easily, we are using the *contextual* package in R. The documentation for this package can be found here. This package allows for easy simulation and implementation of various multi-armed bandit algorithms. For each simulation, one needs to define: (1) a **bandit**, which draws the rewards per arm, (2) a **policy/algorithm** for pulling arms of the bandit, and (3) a **simulator**, which simulates the performance of the policy/algorithm given the bandit.

Below, we go over an example of how one can use the package. Let's start by simply applying the Thompson Sampling algorithm to the ZOZO data. We (1) define a bandit, (2) a policy/algorithm and (3) an agent.

```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
```

```r
#
#   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 = click ~ item_id,
#   no randomization

bandit_Zozo_10 <- OfflineReplayEvaluatorBandit$new(formula = click ~ item_id,
                                                    data = dfZozo_10,
                                                    randomize = FALSE)

# lets generate 10 simulations, each of size 100.000
size_sim <- 100000
n_sim<- 10

# here we define the Thompson Sampling policy object
TS          <- ThompsonSamplingPolicy$new()

# the contextual package works with 'agent' objects - which consist of a policy and a bandit
agent_TS_zozo_10 <-  Agent$new(TS, # add policy
                               bandit_Zozo_10) # add 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(agent_TS_zozo_10, # set our agent
                                      horizon= size_sim, # set the size of each simulation
                                      do_parallel = TRUE, # run in parallel for speed
                                      simulations = n_sim, # simulate it n_sim times
                                    )


# run the simulator object
history_TS_zozo_10            <- simulator$run()

# gather results
df_TS_zozo_10 <- history_TS_zozo_10$data %>%
  select(t, sim, choice, reward, agent)
```

Now that we have gathered the results, let's dive a little bit deeper into each component. First, the bandit object, *OfflineReplayEvaluatorBandit*. The idea behind this function is that it 'replays' the results of an algorithm/policy based on random data. Once an algorithm picks an arm, for instance arm 1, it takes the first observation of arm 1 in the dataset. If the algorithm again picks arm 1, it again samples the next observation from arm 1, etc.

Let us calculate per simulation (1-10), the maximum number of observations.

```r
df_TS_zozo_10_max_t <- df_TS_zozo_10%>%
  group_by(sim) %>% # group by per agent
  summarize(max_t = max(t)) # get max t

df_TS_zozo_10_max_t
```

Task 2: answer below: why is the maximum number of observations different for each simulation?

Your answer to task 2 here:

In the dataframe *df_TS_zozo* we have gathered the results of the TS policy. This dataframe contains the following columns:

- *t*: the step at which a choice was made
- **sim**: the simulation for which we observe results
- **choice**: the choice made by the algorithm
- **reward**: the reward observed by the algorithm
- **agent**: column containing the name of the agent

Task 3: using this dataframe, make a plot of the average cumulative rewards for all simulations, together with the 95% confidence interval.

```r
# Max of observations.
# You may want to adjust this depending on the number of observations per simulation and the size of th
max_obs <- 400

# dataframe transformation
df_history_agg <- df_TS_zozo_10 %>%
  group_by( sim) %>% # group by simulation
  mutate(cumulative_reward = cumsum(reward))%>% # calculate, per sim, cumulative reward over time
  group_by( t) %>% # group by timestep
  summarise(avg_cumulative_reward = mean(cumulative_reward), # average cumulative reward
            se_cumulative_reward = sd(cumulative_reward, na.rm=TRUE)/sqrt(n_sim)) %>% # SE + Confidence
  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 the following two plots:
# 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.
```

As stated at the start, we only have 10 arms for the ZOZO data. However, there are also versions of the data with 20 arms. This is loaded for you below.

Task 4: repeat the simulation steps in the tutorial above, but now for the data with 20 arms.

```r
# this version contains 20 arms
dfZozo_20 <- read.csv('zozo_noContext_20items_tiny.csv')%>%
  select(item_id, click)
```

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

# TODO: Adjust the simulation code in this chunk to fit the data with 20 arms

# Define the bandit
bandit_Zozo_20 <- OfflineReplayEvaluatorBandit$new(formula = click ~ item_id,
                                                   data = dfZozo_10,
                                                   randomize = FALSE)

# define the size of the simulation, as well as the number of simulations
size_sim <- 100000
n_sim <- 10

# define the agent for zozo data with 20 arms
agent_TS_zozo_20 <-  Agent$new(TS, bandit_Zozo_10)

# simulate the results for 10 and 20 arms
simulator          <- Simulator$new(list(agent_TS_zozo_10,
                                         agent_TS_zozo_20),
                                    horizon= size_sim,
                                    do_parallel = TRUE,
                                    simulations = n_sim,
                                    progress_file = FALSE)


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

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

Task 5: Make a plot that compares the average cumulative reward (with 95% confidence interval) for 10 and 20 arms. For which version does Thompson Sampling perform better?.}

```r
# set max observations to create fair comparison across simulations
# You may want to adjust this depending on the number of observations per simulation and the size of th
max_obs <- 200

# set the agent variable to number of
df_TS_zozo$agent <- recode(df_TS_zozo$agent, 'ThompsonSampling' = '10', 'ThompsonSampling.2'='20')
df_history_agg <- df_TS_zozo %>%
  group_by(agent, sim)%>% # group by agent, simulation
  mutate(cumulative_reward = cumsum(reward))%>% # calculate cumulative reward
  group_by(agent, t) %>% # group by agent, timestep t
  summarise(avg_cumulative_reward = mean(cumulative_reward), # calculate average cumulative reward
            se_cumulative_reward = sd(cumulative_reward, na.rm=TRUE)/sqrt(n_sim)) %>% # calculate SE + 
  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 the following two plots:
# 1: A plot that compares the average cumulative rewards over time for 10 and 20 arms using the df_hist
# 2: The plot as defined in (1) together with the 95\% confidence interval.
```