Tutorial 3 - Contextual bandits

Floris Holstege, Bernhard van der Sluis

January 2024

All previous algorithms that we considered were only based on the observed past rewards, and the weighing of uncertainty around those. But obviously, there is a myriad of other forms of information that we can use to assess which arm will yield the most reward. A contextual multi-armed bandit uses this information (referred to as context) to decide which arm to pick at each timestep. This tutorial is about a contextual version of the UCB algorithm called linear UCB, and it is illustrated with an example of cryptocurrency markets.

Theory: contextual UCB

All notation is the same as in the previous tutorial. Recall the UCB method for selecting an arm a_t:

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

where $U_t(a) = \sqrt{\frac{log(t)}{N_t(a)}}$, and $Q_t(a) = \frac{R_1 + R_2 + \ldots + R_{N_t(a)}}{N_t(a)}$. In this case, our estimated $Q_t(a)$ is just based on the sample average.

Now that we want to use context to determine our $Q_t(a)$. With the linear UCB algorithm, we aim to model a linear relationship between contextual features and the reward per arm. Let $\mathbf{X}_{a,t}$ be an $N_t(a)$ x m matrix, with m contextual features for arm a, observed up until moment t. Let $\mathbf{R}_{a,t}$ be a column vector containing the $N_t(a)$ previous rewards for action a, and $\mathbf{x}_{a,t}$ a specific row vector of user features at t for action a. Linear UCB estimates $Q_t(a)$ as follows:

$$Q_t(a) = \arg\max_a [\mathbf{x}'_{a,t} \hat{\boldsymbol{\beta}}_a + c\sqrt{\mathbf{x}'_{a,t} \mathbf{A}^{-1}_{a,t-1} \mathbf{x}'_{a,t}}], \tag{2}$$

$$\mathbf{A}_{a,t-1} = \mathbf{X}'_{a,t-1} \mathbf{X}_{a,t-1}, \tag{3}$$

$$\hat{\boldsymbol{\beta}}_a = \mathbf{A}^{-1}_{a,t-1} \mathbf{X}'_{a,t-1} \mathbf{R}_{a,t-1}. \tag{4}$$

Let's unpack several parts of this equation

$$Q_t(a) = \arg\max_a [\overbrace{\mathbf{x}'_{a,t} \hat{\boldsymbol{\beta}}}^{\text{Est. reward}} + c \overbrace{\sqrt{\mathbf{x}'_{a,t} \mathbf{A}^{-1}_{a,t-1} \mathbf{x}'_{a,t}}}^{\text{Uncertainty}}] \tag{5}$$

- The estimated reward for an arm a is based upon a linear regression between the contextual features ($\mathbf{X}_{a,t-1}$) and the rewards ($\mathbf{R}_{a,t-1}$) up until that point. Using the estimated coefficients of this regression ($\hat{\boldsymbol{\beta}}_a$), we estimate our expected reward for a new observation at t.
- The uncertainty is measured with the standard deviation of the reward for an arm, $\sqrt{\mathbf{x}'_{a,t} \mathbf{A}^{-1}_{a,t-1} \mathbf{x}'_{a,t}}$.

Note that by including an intercept in $\mathbf{X}_{a,t-1}$, one still can incorporate the information from the average reward of pulling an arm. Overall, the linear UCB algorithm is simply an extension of the UCB algorithm that, in addition to the average reward, incorporates information from contextual features.

Example of usefulness of context: trading cryptocurrencies

To illustrate the usefulness of adding context, let's compare the UCB algorithm with the linear UCB algorithm for the purpose of trading cryptocurrencies. You are given data in the 'df_cryptocurrencies.csv' file. This file contains the following information:

- **Symbol**: reflects for which cryptocurrency the information is recorded
- **Returns**: the returns for hold that cryptocurrency at the specific day
- **Date**: day and time for which the returns are recorded
- **OBV**: On-Balance Volume (OBV), is a measure for changes in the the amount of purchases (volume) for a cryptocurrency. Let P_t denote the price at t and Volume_t denote the volume at t. OBV then is calculated as follows:

$$\text{OBV}_t = \begin{cases} \text{if } P_{t-1} < P_t & OBV_{t-1} + \text{Volume}_t \\ \text{if } P_{t-1} \geq P_t & OBV_{t-1} - \text{Volume}_t \end{cases} \tag{6}$$

- **roll returns week**: average returns over the last 5 trading days (a week)
- **prev returns**: returns of the coin for the previous trading day

```
# Packages required for subsequent analysis. P_load ensures these will be installed and loaded.
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
## Warning: package 'pacman' was built under R version 4.3.2
```

```
pacman::p_load(knitr,
               ggplot2,
               png,
               tidyverse,
               rlist,
               contextual,
               lubridate,
               zoo,
               roll)

# get the dataframe with coins
df_coins = read.csv('df_cryptocurrencies.csv')
```

Question 1: run the UCB algorithm (without context) and the Linear UCB algorithm. For the linear UCB algorithm, use the variables *roll returns, prev returns, obv*. The two agents have already been initialized, you only have two define two *OfflineReplayEvaluatorBandit* functions. What arguments do you have to put in the *OfflineReplayEvaluatorBandit* function to get a realistic comparison?

```
coins_bandit  # TODO: add the OfflineReplayEvaluatorBandit function here for UCB algorithm
coins_bandit_context  # TODO: add the OfflineReplayEvaluatorBandit function here for linear UCB algorit

alpha=0.1
UCB <- LinUCBDisjointPolicy$new(alpha=alpha)
linUCB <- LinUCBDisjointPolicy$new(alpha=alpha)
agent <- Agent$new(UCB, coins_bandit, name='UCB')
agent_lin <- Agent$new(linUCB, coins_bandit_context, name='Linear UCB')

?OfflineReplayEvaluatorBandit
# simulate
```

```
size_sim=100000
n_sim=5
simulator          <- Simulator$new(list(agent, agent_lin), # 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,

)


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

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

Question 2: check per sim, per agent, how many observations there are in the simulation. Why does it differ? What does this mean for when you are assessing the differences between agents?

```
# check how many obs
max_obs_per_sim = # TODO: how many observations per simulation, per agent?
```

Question 3: make a plot of the cumulative reward (CR) over time. The cumulative reward should be calculated as:

$$\text{CR}_T = \sum_{t=1}^{T} log(1 + r_t) \tag{7}$$

Also plot the 95% confidence interval. For how many observations should you make the cumulative reward? and which of the two algorithms performs best?

```
# TODO: plot the  95\% confidence interval
```

Question 4: can you make the linear UCB algorithm even better by adding context? create a variable of your own, and assess if it makes the algorithm perform better. Explain why you added the variable.