Published in Towards Data Science

Yogesh Narang Follow
Apr 18, 2020 · 6 min read · Listen

Save

RECOMMENDER SYSTEMS

Recommender systems using LinUCB: A contextual multi-armed bandit approach

Analysis of a contextual multi-armed bandit approach to recommender systems using the disjoint LinUCB algorithm to maximize user interaction

What is the Multi-Armed Bandit Problem?

A multi-armed bandit problem, in its essence, is just a repeated trial wherein the user has a fixed number of options *(called arms)* and receives a reward on the basis of the option he chooses. Say, a business owner has 10 advertisements for a particular product and has to show one of the advertisements on a website. The reward is translated by observing whether the advertisement was **lucrative** enough for the user to **click** on it and get redirected to the product website.

The business owner runs an algorithm for the first 1000 users to decide the best advertisement out of the 10 available advertisements and after his trial run ends, decides to display the **best advertisement** to the rest of the users. The algorithm evaluates the best advertisement on the basis of the performance of the advertisements in the trial run *(First 1000 users)*

Here's where we start thinking. **Is a SINGLE advertisement really good enough to satisfy a vast and diverse audience?**

This is where **contextual** bandits come in. If we knew enough about the user, we could predict with much more accuracy the advertisement that would be best suited towards the user and this is what contextual MAB algorithms do. The advertisement(arm) is selected based on the features of the user (called *context*). Before we move on to analysing one of such algorithms, there's something we need to ponder upon.

Exploration VS Exploitation

> *The dilemma of choosing between exploration and exploitation exists in various aspects of life.*

Say, you go to the ice-cream parlour around the corner and you get your favourite flavour — *chocolate*. You don't try other flavours because you're afraid you might not like them. But, there's also a small probability that if you try a new flavour, say *red velvet*, you might end up liking it even more than chocolate. It's important to **strike the balance** between trying out new flavours (*exploration*) and always getting your favourite (*exploitation*).

MAB algorithms have to find the exact trade-off between choosing a random arm (*which might end up being the optimal arm*) or exploiting its history to choose what it thinks is the best arm (*which might just be a sub-optimal arm because the optimal arm may not have been explored enough yet*)

Online recommender systems try to use such algorithms to display content that they think the user will prefer based on a record of the user's activity on the website.

One such algorithm that we will later build on is called the *upper confidence bound algorithm*.

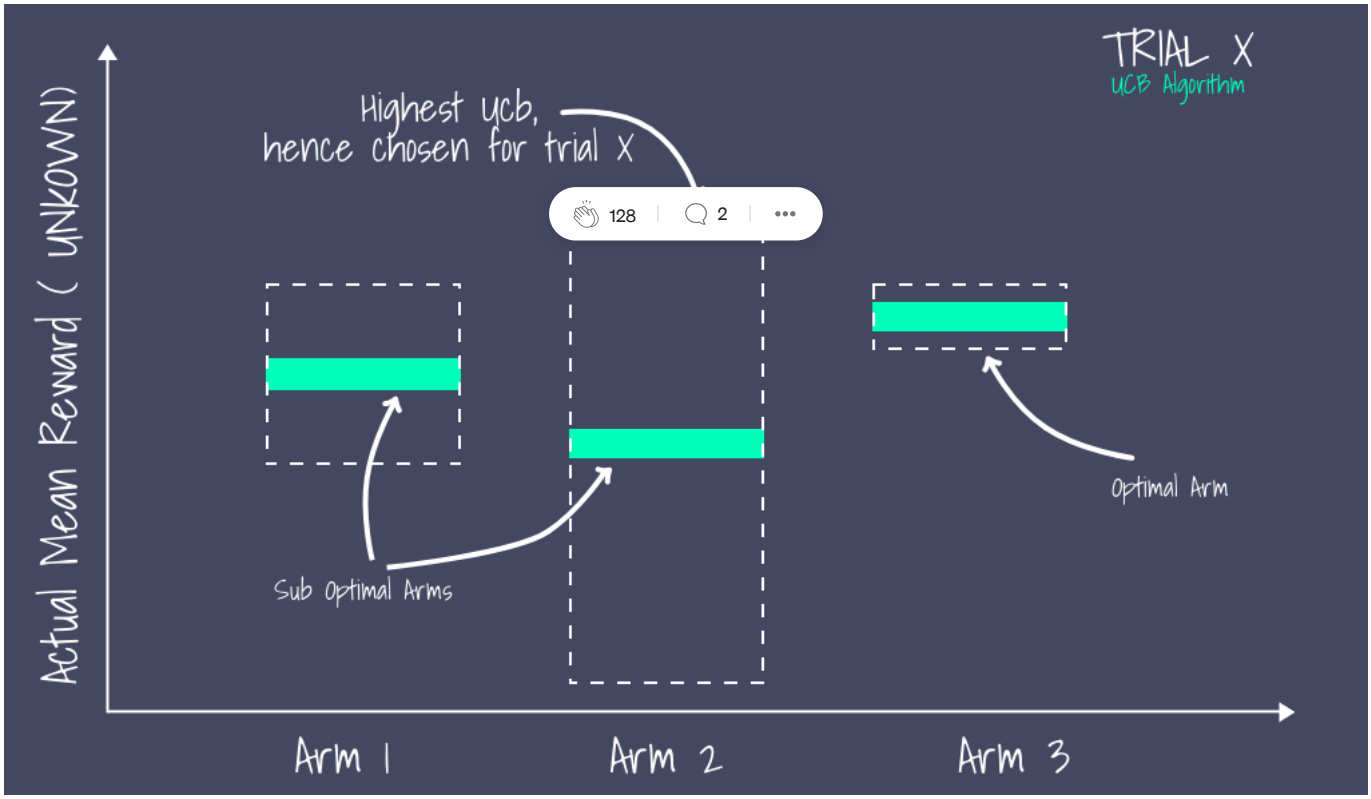
The UCB Algorithm

A very naive greedy approach to solving the multi-armed bandit problem would be selecting the arm that has given us the maximum mean reward with ties being broken arbitrarily. Though this approach tries to select the best possible arm, the algorithm loses and scope for *exploration* and might select a sub-optimal arm in the long run because there can be arms which have not been tried enough

The UCB algorithm keeps a track of the mean reward for each arm up to the present trial and also calculates the upper confidence bound for each arm. The upper bound indicates the ***uncertainty in our evaluation*** of the potential of the arm.

The algorithm is *highly* unsure of an arm's potential if it has a very high upper confidence bound and hence chooses the arm because of a great exploration opportunity.

Consider a running UCB algorithm with its present confidence bounds *(dotted outlines)* as shown in the figure below.



Representation of UCB Algorithm for any Trial X.

Even though the recorded mean award achieved for Arm 3 is higher, the algorithm selects arm 2 because of the *uncertainty of its potential* and updates its confidence bound for future trials.

The UCB algorithm does not take into account u*ser and content features (context)* that may include the user's historical activities at an aggregated level as well as declared demographic information.

LinUCB

The exploitation/exploration problem for contextual MAB problems is formalised as follows :

Formally, a contextual-bandit algorithm A proceeds in discrete trials $t = 1, 2, 3, \ldots$ In trial t:

1. The algorithm observes the current user u_t and a set $\mathcal{A}_t$ of arms or actions together with their feature vectors $\mathbf{x}_{t,a}$ for $a \in \mathcal{A}_t$. The vector $\mathbf{x}_{t,a}$ summarizes information of *both* the user u_t and arm a, and will be referred to as the *context*.
2. Based on observed payoffs in previous trials, A chooses an arm $a_t \in \mathcal{A}_t$, and receives payoff r_{t,a_t} whose expectation depends on both the user u_t and the arm a_t.
3. The algorithm then improves its arm-selection strategy with the new observation, $(\mathbf{x}_{t,a_t}, a_t, r_{t,a_t})$.

Source: https://arxiv.org/pdf/1003.0146.pdf

The expected payoff of an arm is assumed to be linear in its d-dimensional feature vector X with some unknown coefficient vector theta.

$$\mathbf{E}[r_{t,a} | \mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^{\top} \boldsymbol{\theta}_a^*.$$

Source: https://arxiv.org/pdf/1003.0146.pdf

This model is called *disjoint* since the parameters are not shared among different arms. To solve for the coefficient vector theta in the above equation ridge regression is applied to the training data.

An upper confidence bound has to be calculated for each arm for the algorithm to be able to choose an arm at every trial. The strategy for choosing the arm at every trial *t* is formalised as :

$$a_t \stackrel{\text{def}}{=} \arg\max_{a \in \mathcal{A}_t} \left(\mathbf{x}_{t,a}^{\top} \hat{\boldsymbol{\theta}}_a + \alpha \sqrt{\mathbf{x}_{t,a}^{\top} \mathbf{A}_a^{-1} \mathbf{x}_{t,a}} \right)$$

Source: https://arxiv.org/pdf/1003.0146.pdf

The goal of the algorithm is to maximise total reward (total user clicks in the long run)

The algorithm is formalised by its authors as follows :

Algorithm 1 LinUCB with disjoint linear models:

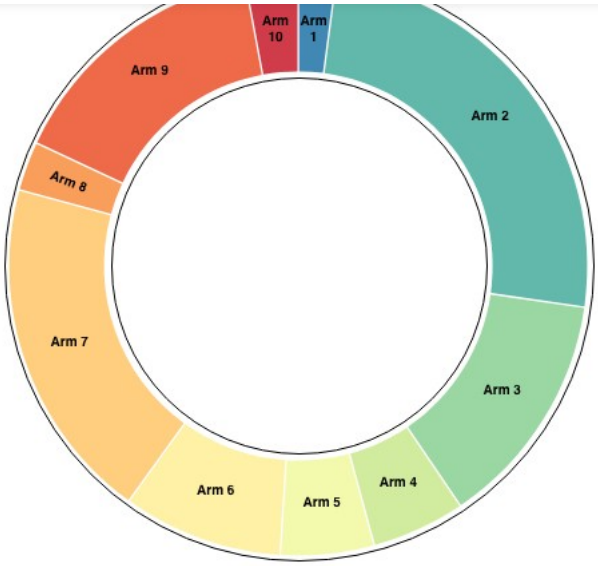
Inputs: $\alpha \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d \times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1} \mathbf{b}_a$
    $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a \in \mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t} \mathbf{x}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{t,a_t}$
end for

Source: https://arxiv.org/pdf/1003.0146.pdf

We don't dwell upon the mathematics of the algorithm but we focus on the results that I achieved by simulating the algorithm on the standard dataset.

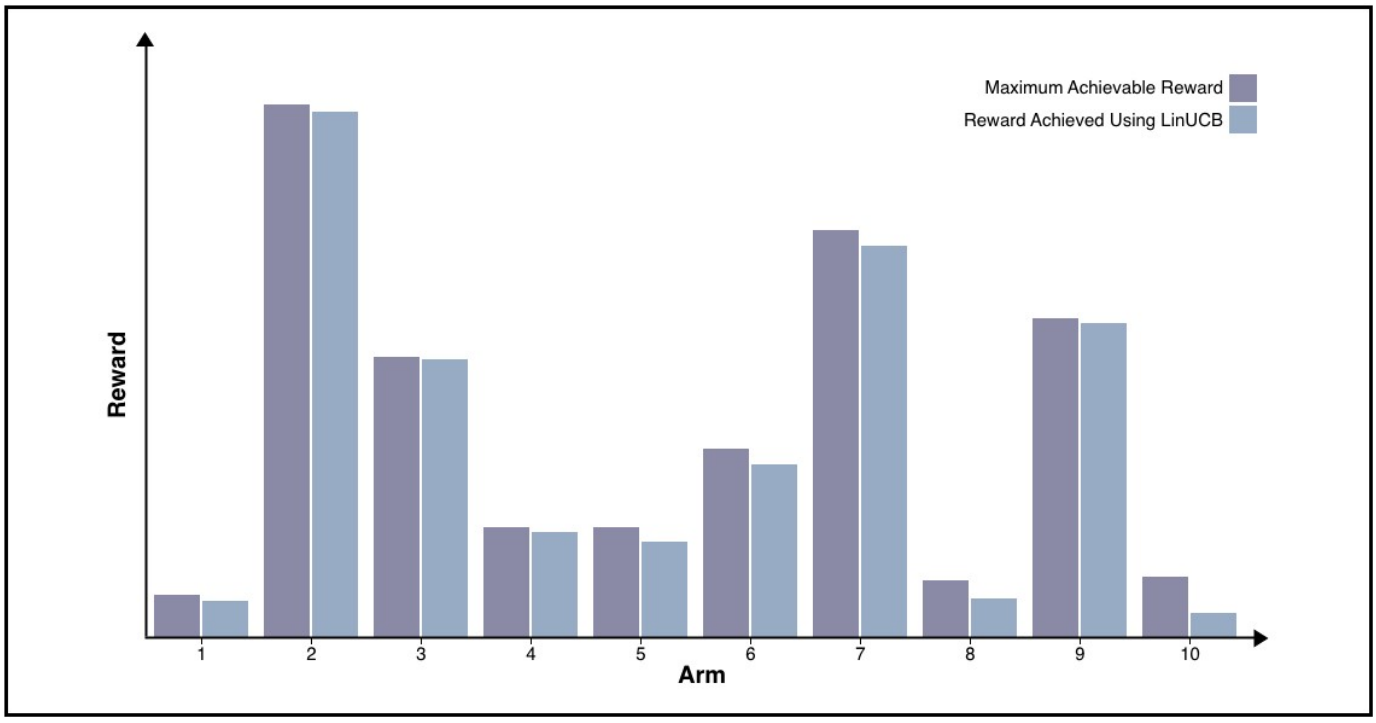
Usually, MAB problems are analysed on the basis of **regret** that is the difference of the total reward achieved by always selecting the optimal arm and the total reward achieved by the algorithm.

The below doughnut plot shows a comparison of the maximum reward we can achieve if we choose only one arm (and treat it as the optimal arm). The plot indicates that irrespective of the chosen optimal arm, if we don't show flexibility in our decisions, we can achieve a maximum of 20% of the total reward. This type of comparison isn't typically possible in real life because we don't have access to the entire data beforehand, but this helps us realise the fact that we can't just rely on one optimal arm.



A comparison of the total reward achieved when each arm was individually selected as the optimal arm and chosen for each trial

The LinUCB algorithm does an excellent job of deciding the choice for the arm in every trial and the below plot reflects the same. It compares the maximum possible achievable reward per arm to the reward actually realised per arm. The results indicate that we're able to exploit each arm up to 90% of its potential.



Maximum achievable reward vs. reward achieved using LinUCB for each arm

Recommender Systems can be directly modelled as a contextual MAB problem where the different options for recommendations are the arms and whether the user likes the recommendation can be translated to the reward and the ultimate goal is to be able to provide personalised recommendations to each user.

CMAB algorithms are combined with various filtering techniques based on user preferences to achieve an even more personalised recommendation.

SUMMARY

REFERENCES

- Sutton, Richard S. and Barto, Andrew G., Reinforcement learning — An introduction http://incompleteideas.net/book/the-book-2nd.html
- A Contextual-Bandit Approach to Personalized News Article Recommendation https://arxiv.org/pdf/1003.0146.pdf

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get this newsletter Emails will be sent to carloghiglione@virgilio.it.
Not you?