LinUCB vs HybridLinUCB in recommender systems

Author:
Radek Bartyzal (bartyrad@fit.cvut.cz)

April 5, 2018

1 Introduction

The algorithms LinUCB and HybridLinUCB are both introduced in the article A Contextual-Bandit Approach to Personalized News Article Recommendation published in 2012.

The authors decided to model personalized recommendation of news articles as a contextual bandit problem, a principled approach in which a learning algorithm sequentially selects articles to serve users based on contextual information about the users and articles, while simultaneously adapting its article-selection strategy.

2 Theory

Both algorithms are based on upper confidence bound (UCB) method that uses a smarter way to balance exploration and exploitation than a simple $\epsilon - greedy$ strategy. Specifically, in trial t, these algorithms estimate both the mean payoff $\mu_{t,a}$ of each arm a as well as a corresponding confidence interval $c_{t,a}$, so that $|\mu_{t,a} - \mu_a| < c_{t,a}$ holds with high probability. They then select the arm that achieves a highest upper confidence bound: $a_t = argmax_a(\mu_{t,a} + c_{t,a})$. With appropriately defined confidence intervals, it can be shown that such algorithms have a small total T- trial regret that is only logarithmic in the total number of trials T, which turns out to be optimal.

2.1 LinUCB with Disjoint Linear Models (LinUCB)

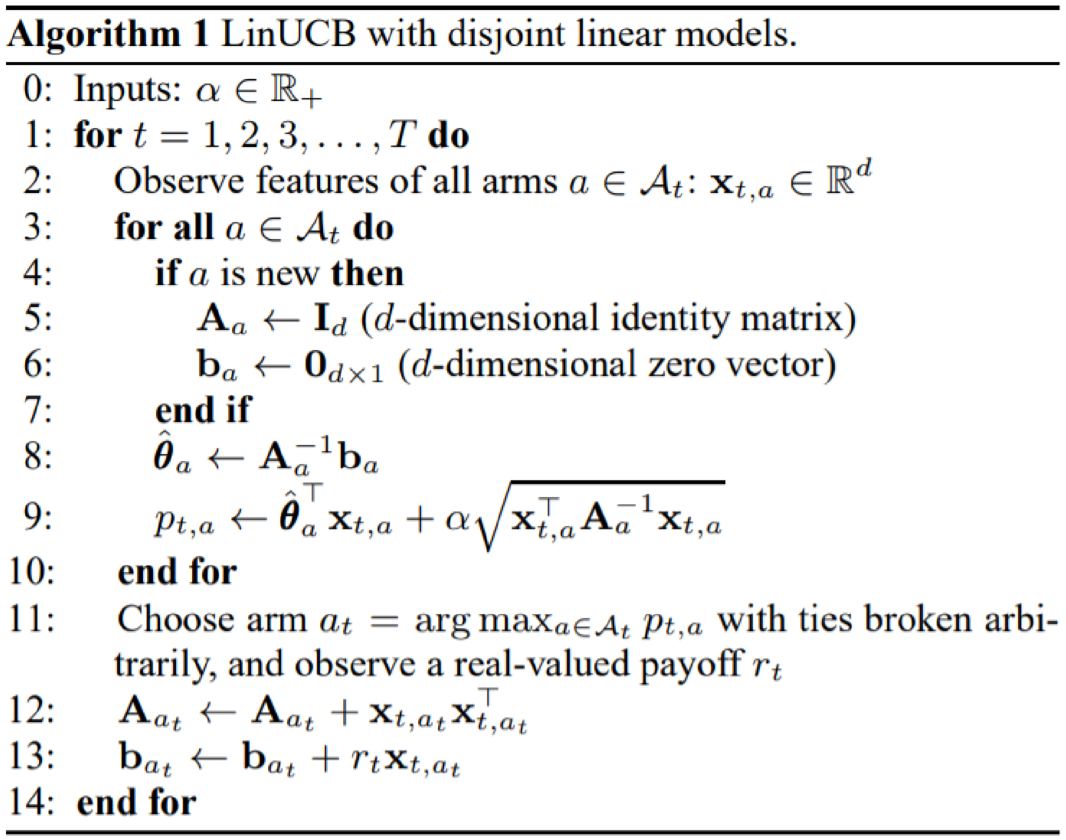
We assume the expected payoff of an arm a is linear in its d-dimensional feature $x_{t,a}$ with some unknown coefficient vector θ^*_a, namely:

$$E[r_{t,a}|x_{t,a}] = x^T_{t,a}\theta^*_a$$

- a = arm = action = item to be recommended = e.g. article
- t = trial = in this case user ID
- $r_{t,a}$ = reward of action a in trial t
- $x_{t,a}$ = features describing **both** user and the selected article a at trial t. If the features of a user are his ratings, they will change with time.

- θ_a^* = an unknown coefficient vector specific to arm a

This model is called disjoint since the parameters are not shared among different arms. The solution is reached by a simple ridge regression and an algorithm allowing incremental updating of the parameter matrices can be seen in Figure 1.



Algorithm 1 LinUCB with disjoint linear models.

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

Figure 1: LinUCB algorithm with incrementally updating parameter matrices.[1]

2.2 LinUCB with Hybrid Linear Models (HybridLinUCB)

In many applications, it is helpful to use features that are shared by all arms, in addition to the arm-specific ones. For example, in news article recommendation, a user may prefer only articles about politics for which this provides a mechanism. Hence, it is helpful to have features that have both shared and non-shared components. Formally, we adopt the following hybrid model by adding another linear term to the previous equation:

$$E[r_{t,a}|x_{t,a}] = z_{t,a}^T\beta^* + x_{t,a}^T\theta_a^*$$

- $z_{t,a}$ = features of the current user/article combination
- β^* = an unknown coefficient vector common to all arms

The algorithm can be seen in Figure 2.

3 Implementation

I have implemented both algorithms using the paper's notation for easy readability. The implementation does not support changing the arm set dynamically at the moment which leads to increased performance. The full project implementation can be found at [2].

Features in $x_{t,a}$ are a concatenation of features of user t and item a at the current time.

- user features = his ratings of all items = R_u
- item features = movie genres = array of 1/0 describing whether the movie belongs to a genre. There are 19 genres.
- $x_{t,a}$ = user features concatenated with item features
- $z_{t,a}$ = only article features = genre vector

The algorithms are always run for a selected number of epochs. An epoch means that the algorithm has iteratively generated a single recommendation for each user in the dataset. The ordering of the users is random in each epoch.

When there are multiple possible arms to choose from = their p_t equals $max(p_t)$, final arm is selected randomly from such arms.

Algorithm 2 LinUCB with hybrid linear models.

```
0: Inputs: $\alpha \in \mathbb{R}_+$
1: $\mathbf{A}_0 \leftarrow \mathbf{I}_k$ ($k$-dimensional identity matrix)
2: $\mathbf{b}_0 \leftarrow \mathbf{0}_k$ ($k$-dimensional zero vector)
3: for $t = 1, 2, 3, \ldots, T$ do
4:     Observe features of all arms $a \in \mathcal{A}_t$: $(\mathbf{z}_{t,a}, \mathbf{x}_{t,a}) \in \mathbb{R}^{k+d}$
5:     $\hat{\boldsymbol{\beta}} \leftarrow \mathbf{A}_0^{-1}\mathbf{b}_0$
6:     for all $a \in \mathcal{A}_t$ do
7:         if $a$ is new then
8:             $\mathbf{A}_a \leftarrow \mathbf{I}_d$ ($d$-dimensional identity matrix)
9:             $\mathbf{B}_a \leftarrow \mathbf{0}_{d\times k}$ ($d$-by-$k$ zero matrix)
10:            $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ ($d$-dimensional zero vector)
11:        end if
12:        $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\left(\mathbf{b}_a - \mathbf{B}_a\hat{\boldsymbol{\beta}}\right)$
13:        $s_{t,a} \leftarrow \mathbf{z}_{t,a}^\top \mathbf{A}_0^{-1}\mathbf{z}_{t,a} - 2\mathbf{z}_{t,a}^\top \mathbf{A}_0^{-1}\mathbf{B}_a^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a} +$
           $\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{B}_a\mathbf{A}_0^{-1}\mathbf{B}_a^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}$
14:        $p_{t,a} \leftarrow \mathbf{z}_{t,a}^\top\hat{\boldsymbol{\beta}} + \mathbf{x}_{t,a}^\top\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{s_{t,a}}$
15:    end for
16:    Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbi-
       trarily, and observe a real-valued payoff $r_t$
17:    $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
18:    $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
19:    $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
20:    $\mathbf{B}_{a_t} \leftarrow \mathbf{B}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{z}_{t,a_t}^\top$
21:    $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
22:    $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{z}_{t,a_t}\mathbf{z}_{t,a_t}^\top - \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
23:    $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + r_t\mathbf{z}_{t,a_t} - \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
24: end for
```

Figure 2: HybridLinUCB algorithm with incrementally updating parameter matrices.[1]

4 Experiments

4.1 Data preprocessing

I have decided to use the MovieLens 100k dataset. It has 100 000 ratings, 1000 users and 1700 movies. [3]

Due to the computational requirements I have taken a small subset of the whole dataset containing 100 items and 56 users. To ensure that all users have at least some ratings I have randomly added 3 ratings to each user.

As is customary in the recommendation world I have binarized the ratings from a scale of 1-5 to:

- 1 if the rating is 4 or larger = positive rating
- -1 if the rating is smaller than 4 = negative rating
- 0 = unknown rating

I have decided to keep the negative ratings separated from the unknown ones because I will need them to model the unknown user ratings.

4.2 Modeling the user behavior

The paper describes a sophisticated method to evaluate the algorithms offline which are interesting nevertheless such a procedure is out of scope of this project. Therefore I have decide to use a simple user model predicting a positive or negative rating of a yet unseen item.

Let item a be recommended to user u. If the user u has already rated the item a, the returned reward will be 1 for positive rating or 0 for negative rating. Instead of these fixed values a Bernoulli distribution with high probability of 1/0 can be used to calculate the reward.

If the user u has not rated the item i yet the reward will be sampled from a Bernoulli distribution with p equal to how much the user likes the genre of item i. Likability of a genre g_a is calculated as a ratio of positive ratings of items belonging to genre g_a to a number of negative ratings of items belonging to g_a. Only ratings by user u are counted. If item i belongs to multiple genres the resulting likability is calculated as an average of all the item i genre likabilities.

4.3 Experiment 1: Fixed rewards

In this experiment I have simply run the implemented algorithm with settings described in the previous chapters for 50 epochs. The results can be seen in Figures 3 and 4.

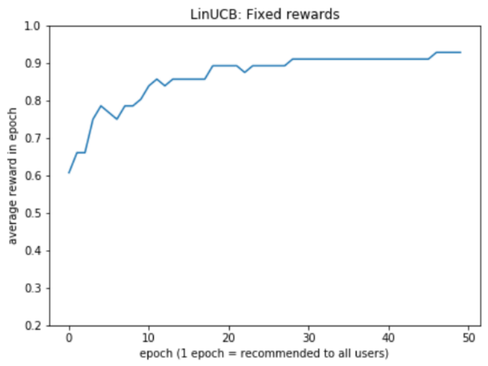


Figure 3: LinUCB with fixed rewards for already rated items.

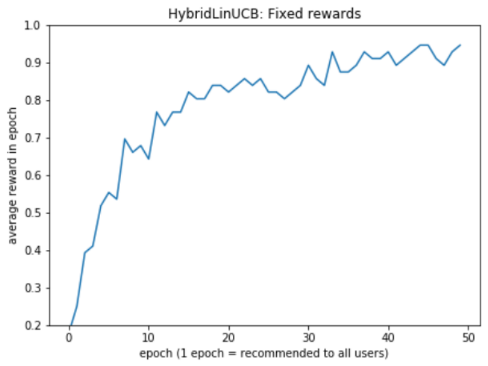


Figure 4: HybridLinUCB with fixed rewards for already rated items.

As can be observed LinUCB quickly reaches the exploitation phase which is especially easy since it just needs to find 1 item that returns positive reward for each user and then select it all the time.

HybridLinUCB on the other hand is exploring more and therefore has a slower increase in average reward per epoch, but achieves similar results in the end.

4.4 Experiment 2: Stochastic rewards

In this experiment I have modified the user model behavior when giving reward for already rated items. If the previous rating was positive the reward will be sampled from Bernoulli distribution with $p = 0.9$, otherwise $p = 0.1$. All the other parameters remained the same. The results can be seen in Figures 5 and 6.

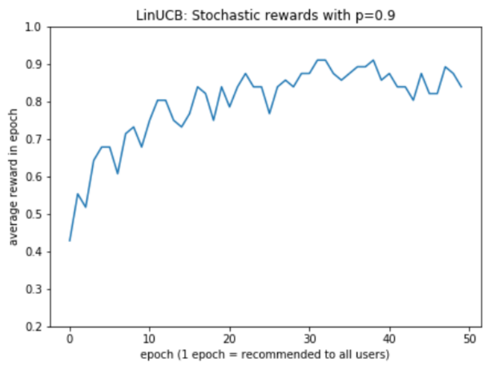


Figure 5: LinUCB with stochastic rewards for already rated items.

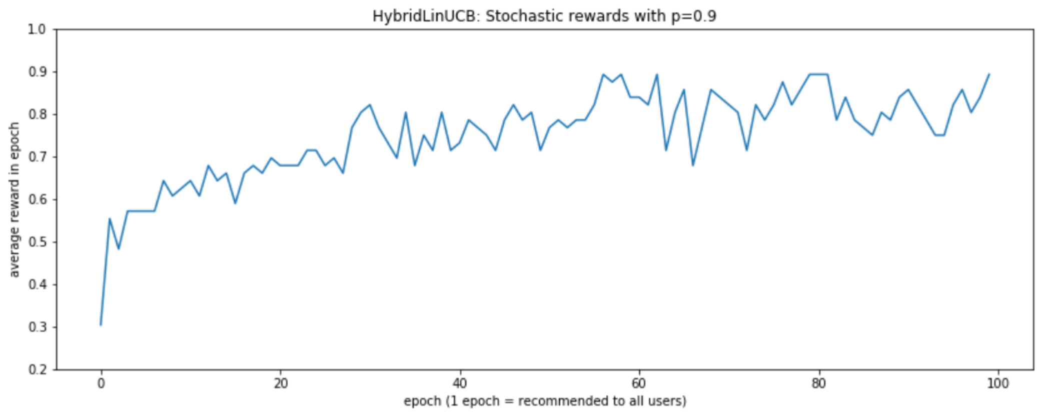


Figure 6: HybridLinUCB with stochastic rewards for already rated items.

LinUCB is again trying to exploit quite quickly but its progress is hindered by the noise, nevertheless it manages to stay around the near optimal 0.85 in the end.

HybridLinUCB is again exploring more and the noisy rewards are slowing it down even more, that is why I let it run for 100 epochs. It can be seen that it ends up oscillating around slightly worse 0.8 average reward.

References

[1] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." Proceedings of the 19th international conference on World wide web. ACM, 2010. Accessible from: https://arxiv.org/abs/1003.0146

[2] Project implementation at: https://github.com/BartyzalRadek/contextual-bandits-recommender

[3] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872 Accessible from: https://grouplens.org/datasets/movielens/100k/