orcs_4200_project

Muti-Armed Bandits RecSys (KuaiRec Dataset)

We are going to use multi-armed bandits to create a recommendation system. We are going to simulate whether users will click on a specific video or not, and based on that make future predictions.

KuaiRec Dataset

The KuaiRec Dataset contains user interactions on a webpage. source

Implementation can be found at bandit_Nov22.ipynb

For this project, we focus on the videos a user clicks. Our goal is to predict which video a user will click next, given the videos they clicked on in the past.

	user_id	video_id	play_duration	video_duration	time	date	timestamp	watch_ratio	first_level_category_id
0	14	148	4381	6067	97462.318	20200705.0	1.593898e+09	0.722103	19.0
1	14	183	11635	6100	97473.997	20200705.0	1.593898e+09	1.907377	28.0
2	14	3649	22422	10867	97543.419	20200705.0	1.593898e+09	2.063311	28.0
3	14	5262	4479	7908	97637.225	20200705.0	1.593898e+09	0.566388	5.0
4	14	8234	4602	11000	97937.399	20200705.0	1.593899e+09	0.418364	6.0
...	...	...	...	...	...	...	...	...	...
4676565	7162	2267	11908	5467	2423337.210	20200801.0	1.596224e+09	2.178160	25.0
4676566	7162	2065	11919	6067	2423337.210	20200801.0	1.596224e+09	1.964562	29.0
4676567	7162	1296	16690	19870	2423337.210	20200801.0	1.596224e+09	0.839960	1.0
4676568	7162	4822	11862	24400	2423337.210	20200801.0	1.596224e+09	0.486148	9.0
4676569	7162	4364	2182	19367	2423337.210	20200801.0	1.596224e+09	0.112666	25.0

4676570 rows × 15 columns

Issues

The dataframe is too sparse (no user has watched every video on the website) and there are too many videos to choose from.

We resolve this by grouping each observation in the user interaction matrix by user, taking the sum of their watch ratio per catigory:

```
small_transformed_df = small_matrix_merged.pivot_table(index='user_id',
                                                        columns='first_level_category_id',
                                                        values='watch_ratio',
                                                        aggfunc='sum')
```

first_level_category_id	-124.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
user_id										
14	4.286082	143.114573	18.780785	3.990782	23.204983	127.466564	226.189900	154.046108	274.076947	104.9
19	2.181404	114.764137	15.067363	4.878045	26.105601	110.512886	184.396166	125.246851	260.217647	91.96
21	2.926339	124.110059	23.230539	5.624282	27.482122	130.194884	240.155861	142.008902	289.964595	93.6
23	3.981672	107.067355	26.635610	6.181167	20.649301	131.889982	191.524655	154.143455	317.001194	124.5
24	2.684146	89.313163	13.462446	5.342142	26.527966	132.614202	178.508812	113.466052	323.619663	91.33

5 rows × 38 columns

Lastly, we merge the features of each user (as one-hot encodings), to the above matrix. This gives the user features and video watch ratios on each row:

```
small_transformed_merged_df = (
    user_features
    .merge(small_transformed_df, on="user_id", how="right")
)
```

	const	onehot_feat0	onehot_feat1	onehot_feat2	onehot_feat3	onehot_feat5	onehot_feat6	onehot_feat7	onehot_feat8	on

user_id	const	onehot_feat0	onehot_feat1	onehot_feat2	onehot_feat3	onehot_feat5	onehot_feat6	onehot_feat7	onehot_feat8	on
user_id										
~~**14**~~	~~1.0~~	~~0~~	~~5~~	~~8~~	~~417~~	~~0~~	~~1~~	~~3~~	~~297~~	~~4~~
~~**19**~~	~~1.0~~	~~0~~	~~1~~	~~18~~	~~589~~	~~0~~	~~1~~	~~7~~	~~227~~	~~3~~
21	1.0	0	4	13	568	0	0	13	292	4
23	1.0	0	1	3	45	0	0	13	148	6
24	1.0	1	4	17	634	0	1	0	64	5

5 rows × 50 columns

Solution: LinUCB

Now we can formally define the problem with LinUCB:

$$
\begin{aligned}
\text{Arms: } & K = 40 && \text{The video catagory} \\
\text{Features: } & x_{t,a} \in \mathbb{R}^{11} && \text{The user features} \\
\text{Reward Function: } & Reward{t,a_t} = x^\top{t,a}\beta_a + \epsilon_{t,a} \\
\text{Objective: } & \min_{a_t} Regret_t = && \sum_{t=1}^{T} (Reward_{t,a_t} - Reward_{t,a_t})
\end{aligned}
$$

After training, the cumulative regret looks like the following: