Data Science Project A1 - IASD 2023
A Linear Bandit Approach to Movie Recommendation

Group Max_Hipp_Yann: Maximilien Wemaere, Hippolyte Gisserot, Yann Trémenbert

1 Introduction

1.1 Context

What with the growing demand in dematerialized movies and series accessibility, the competition between major actors such as *Netflix*, *Amazon Prime* or *Disney+*, has contributed to the rising interest in movie recommendation.

In this report, we tackle the applicability of a Linear Bandit approach to such recommendation system as explained in *A contextual-bandit approach to personalized news article recommendation (2012)* [LCLS10]. To this end, we consider the MovieLens 25M Dataset containing 25 million ratings and for a total of 62,000 movies and 162,000 users, along with a matrix of features for each movie (genre, date of release...). From this dataset, we can build a matrix of ratings $R = (r_{i,m})_{i,m}$ between users i and movies m, which is split into two matrices R_{train} and R_{test}. They constitute on the one hand the matrix used to build user and movie features, and on the other hand the matrix used to compare our predictions with the actual rating given by a user on a certain movie. The goal is to guess which movie is indeed the best rated movie among all the movies rated by a single user.

The behavior of a contextual bandit algorithm to the problem of movie recommendation can be formulated as following:
We define a trial as each time a user wants to watch a movie among our recommendations. For each trial $i = 1, 2, 3, ...$

1. The algorithm considers the current user u_i and a set of N_m movies together with their feature vectors i.e. contexts $x_{i,m}$ for each movie.
2. Based on observed payoffs in previous trials, we select a movie m_i, and receive a payoff $r_{i,m_i} \in [0, 5]$ which corresponds to the actual grade given by the user in R_{test}.
3. We improve our movie-selection strategy by taking into account the new observation $(x_{i,m_i}, m_i, r_{i,m_i})$.

We will see that there exists different techniques in order to select a movie based on past observations ranging from simple techniques (random-strategy, ϵ-greedy strategy, K-bandit...) - some of which we will implement for reference - to more complex ones like the contextual or hybrid linear models from [LCLS10]. Usually, these techniques rely on the concept of Upper Confidence Bound (UCB): The algorithm will not only evaluate the expected payoff of a movie, but will also calculate confidence bounds i.e. the associated uncertainty of the expected payoff. Thus, the algorithm will not select the movie with the highest expected payoff, but the one which *potentially* achieves the maximum reward as depicted in the following figure:

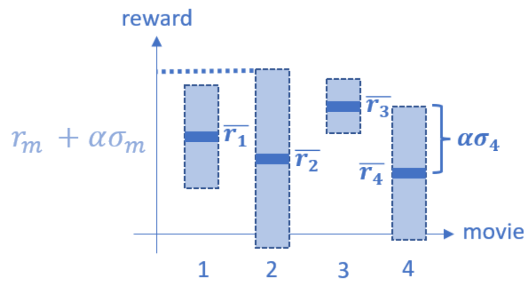

Figure 1: UCB selection strategy

In order to compare our different movie selection strategies, we define the regret $Regret_i = r^* - r_{chosenmovie}$ as the difference between the rating of the best possible movie rated by the user i and the rating obtained with our strategy. This definition allows us to calculate our reference metrics for comparison, the cumulated regret, defined as $CumRegret_n = \sum_{i=1}^{n} Regret_i$. This cumulated regret, when plotted as a function of iterations, is an increasing function supposed to flatten as the algorithm is learning.

1.2 Notations

Below is a summary of the main notations we use in this report:

- N_m: total number of movies
- n_m: number of movie features
- $M = (x_m)_{1 \leq m \leq N_m} \in \mathbb{R}^{n_m \times N_m}$: matrix/vectors of movie features (release date, genres, constant)
- $(m_i)_{1 \leq i \leq N_{m_i}}$: movies seen by user i
- N_u: total number of users
- n_u: number of user features
- $U = (x_i)_{1 \leq i \leq N_u} \in \mathbb{R}^{n_u \times N_u}$: matrix/vectors of user features
- $(z_{i,m})_{i,m} = (x_i x_m)_{i,m} \in \mathbb{R}^{(N_u \times N_m) \times 1}$: interaction vectors between user and movie features
- $r_{i,m}$: rating given by user i to movie m
- $R = (r_{i,m})_{i,m} \in \mathbb{R}^{N_u \times N_m}$: matrix of ratings
- $R_{train} \in \mathbb{R}^{N_u \times N_m}$: training matrix of ratings, used to compute user features (50% of the whole data set)
- $R_{test} \in \mathbb{R}^{N_u \times N_m}$: test matrix of ratings, used to test the different algorithms
- $R = (r_{i,m})_{i,m} \in \mathbb{R}^{N_u \times N_m}$: matrix of ratings
- r_i: vector of ratings for movies rated by user i
- i_m: number of times movie m was selected over the $i-1$ first iterations
- X_i: matrix of features for movies selected over the $i-1$ first iterations
- $X_{i,m}$: matrix of features for users to which movie m was recommended over the $i-1$ first iterations
- y_i: vector of ratings for movies selected over the $i-1$ first iterations
- $y_{i,m}$: vector of ratings for movie m each time it was selected by a user
- M_i: matrix of features for movies rated by user i

2 Various approaches to solving the recommendation problem

2.1 Random approach

The random approach consists in selecting a random movie among all the movies rated by the user i. This method is obviously not performing well as the cumulated regret is linear (figure 2) i.e. there is no improvement over time.

2.2 Context-free K bandits

One famous strategy is the K-bandit algorithm which calculates the expected payoff $\overline{r_{i,m}}$ as the empirical averaged payoff over previous observations, and its associated confidence bounds $\sigma_{i,m}$ as a decreasing function of the number of selections. Therefore, the chosen movie is:

$$m_i^* = \underset{m \in (m_i)}{\operatorname{argmax}} \, \overline{r_{i,m}} + \frac{\alpha}{\sqrt{i_m}}$$

As highlighted by the orange flattening curve in figure 2, the algorithm is learning and succeeds in recommending movies which better fit the users over time.

2.3 Context-free linear bandits

Let's now develop a new approach. In the previous section, we simply recommended movies based on the upper confidence bound of their empirical average rating. Now, we want to express the expected rating as a linear function of the movie features:

$$\mathbb{E}[r_{i,m}|x_m] = x_m^T \theta_i^*$$

Where θ_i^* is computed the following way, using Ridge regression:

$$\theta_i^* = \underset{\theta_i \in \mathbb{R}^{n_m}}{\operatorname{argmin}} \|X_i\theta_i - y_i\|^2 + \lambda\|\theta_i\|^2$$

Finally, assuming that movie features (x_m) are i.i.d. and drawn from a multivariate normal distribution, we recommend at iteration i the movie with highest expected rating upper confidence bound:

$$m_i^* = \underset{m \in (m_i)}{\operatorname{argmax}}\ x_m^T\theta_i + \alpha\sqrt{x_m^T(X_i^T X_i + \lambda I)^{-1} x_m}$$

Figure 2 shows that this approach is more efficient in the short run but gets outperformed beyond the 3,000th iteration. The algorithm seems to learn quite quickly, but has little room for improvement. This could come from the fact that movie features do not bring more information than the constant coefficient (the average) does. Moreover, we really want now to find a less naive way to recommend movies. So far, we did not take into consideration information about the user preferences and selected movies based solely on their general popularity. Let's explore a new approach in the following section.

2.4 Contextual linear bandits

This new approach aims at incorporating user features in the learning process. Since we do not have access to user features, we need to find a way to compute them in a proper way. Let's express them as a linear function of the movie features and then add a constant column:

$$U^* = \underset{U \in \mathbb{R}^{n_m \times N_u}}{\operatorname{argmin}} \|UM^T - R_{train}\|^2$$

We now want to express the expected rating as a linear function of the user features, not the movie features anymore:

$$\mathbb{E}[r_{i,m}|x_i] = x_i^T \theta_{i,m}^*$$

Therefore, at each iteration i and for each movie m, we solve the following optimization problem:

$$\underset{\theta_{i,m} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \|X_{i,m}\theta_{i,m} - y_{i,m}\|^2 + \lambda\|\theta_{i,m}\|^2$$

Finally, we recommend the movie with highest expected rating upper confidence bound:

$$m_i^* = \underset{m \in (m_i)}{\operatorname{argmax}}\ x_i^T\theta_{i,m} + \alpha\sqrt{x_i^T(X_{i,m}^T X_{i,m} + \lambda I)^{-1} x_i}$$

2.4.1 Using the full range of features

Let's first use the full range of movie features for the computation of matrix U^* and analyse the resulting performance (figure 2). This new approach performs approximately the same as the context-free linear-bandit algorithm and much worse than the context-free K-bandit approach, which is, remember, the most simple and naive non-trivial approach we implemented so far. This poor result might come from the fact that our user features are highly artificial and that we lack some data points to estimate them in a more robust way.

2.4.2 Using a reduced number of features

A solution that could partially remedy the problems mentioned above would be to reduce the number of movie features used to compute user features, let's say, from 21 to 5. To do it, let's rely on both PCA and K-Means clustering. Relying on dimension reduction with PCA manages to improve the performance of our algorithm (figure 2). This might come from the fact that as we highly reduced the number of coefficients to fit, we needed fewer data points. However, it still does not outperform the most naive K-bandit approach, probably still due to the artificial computation of user features.

2.5 Hybrid linear bandits

This last approach aims at combining the information given by movie and user features (whose dimensions are reduced by PCA), to see if we could take advantage of these interactions. This time, we want to express the expected rating at each iteration i and for each movie m the following way:

$$\mathbb{E}[r_{i,m}|x_i, z_{i,m}] = x_i^T \theta_{i,m}^* + z_{i,m}^T \ \beta_i^*$$

NB: β_i^* a coefficient vector which is constant across movies, while $\theta_{i,m}^*$ is not. The idea is that we do not want to fit the entire set of coefficients for each movie. Rather, we want to have some that stay constant and others that vary.

The hybrid approach has about exactly the same performances as the contextual linear-bandit approach (figure 2). It seems that the interactions between movie and user features do not bring much information to the problem. Moreover, this pretty complex approach still shows lower performances than the most naive one. Now, we can maybe reasonably conclude that, with respect to our data set, users do not really have specific preferences about movies, hence the superiority of the context-free non-linear method.

2.6 Output graph

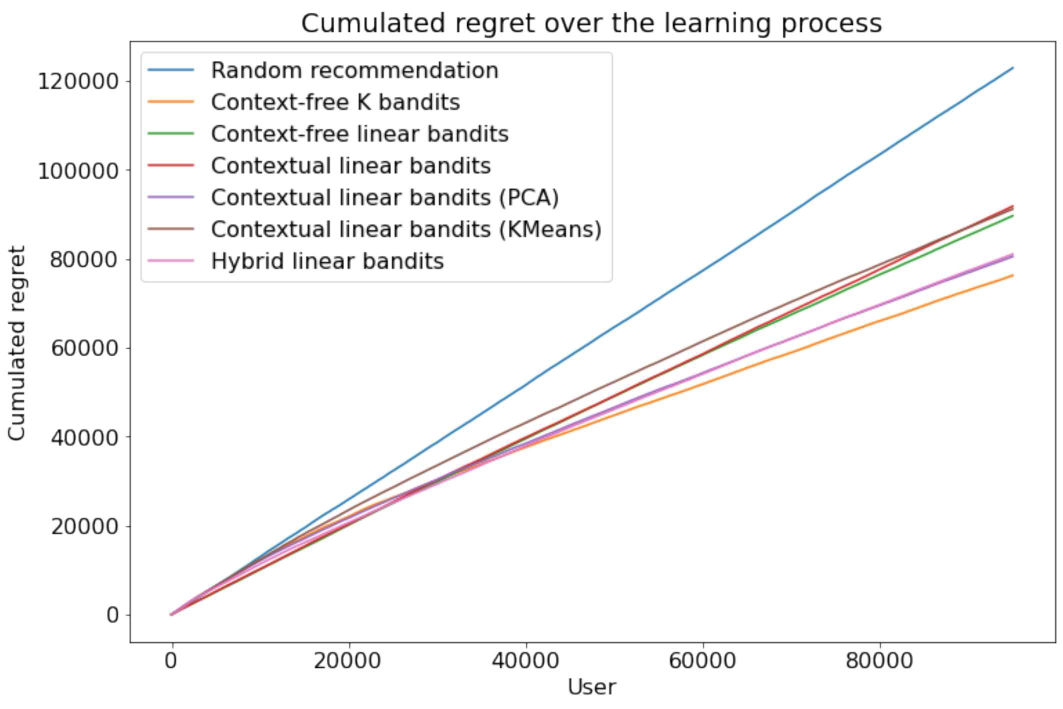

Figure 2: Cumulated regrets over the learning phase for the different approaches ($\lambda = 1$, $\alpha = 2$)

3 A new framework: linear bandits in real time

Finally, we worked on a new approach of the problem, an approach which could be used in a real context. The objective is to be able to learn even if the movie we recommend is not chosen.

At each time t_i a user rates a movie. So after ordering our dataset according to the time, we can browse it, and at each time, we feed our Linear UCB algorithm with the id of the user like in real life when a user is connected to the movie platform.

As seen in the previous part, the algorithm creates a vector of upper-bounds of all movies. We transform this vector into a vector of ratings. We do this by setting the highest upper bound to 5 star, the lowest to 0.5 and all the other rates are determined based on their distance to the extremities. So we have a rate prediction for all movies.

Thus we can recommend the best movie with the highest rating (which is the highest upper-bound, as in LUCB). However, in the dataset as well as in real life, the user does not necessarily look at the movie recommended. Our algorithm stands out here by only comparing all the ratings of the movies truly seen by the user and their predicted ratings (in the vector of ratings) even if it has not been chosen by the algorithm. Then, we can update our LUCB algorithm.

To conclude with this new approach, we saw that contrary to the other ones it can be used in a real conditions and in real time. However, it is computationally very costly and we did not achieve to get good results as shown by the evolution of the score, which is not raising. The possible source of error is the way of predicting the scores, it should be possible to take into account the average score given by the user as well as the distribution of ratings given by the user (there will never be as many 0.5 stars as 5 stars or 3 stars movies for example).

4 Conclusion

We have seen that among all approaches the most naive one works best in that particular setting. Different hypothesis came to us in order to explain this result:

- The originality and uniqueness of the users is drowned in their number and their uniformity prevails in this data set. Which is why recommending movies as a trend is the best strategy.
- Without the users' features we cannot exploit the efficiency of the Linear Bandit, as used in the reference article. This is why we tried to create them from a part of the data set, which could be incomplete or biased.
- The rating matrix was highly sparse and we decided to only consider the most watched movies. So the originality and uniqueness of users could have disappeared.
- The K-bandit algorithm do not use users' features, which is tantamount to considering only one "averaged" user and iterating over him for more than 80,000 iterations, hence the learning process. When considering our created users' features, we could not iterate as much on each user, which could explain why our algorithms perform worse on the first 80,000 iterations.

Finally these methods involve to know the movie which will be watched to rate it. So we can't train the algorithms in real conditions. Thus, the last presented method has been designed to propose a solution usable in real conditions.

For further research, we could implement more features such as the average rating for each movie category or user, or a system of distance between users to propose recommendation based on similarity.

References

[LCLS10] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.