


# Enhancing Recommendation Systems Through Contextual Bandit Models

Zhaoxin Chen <sup>a</sup>

*School of Economics and Management, Xi'an University of Technology, Xi'an, Shaanxi Province, 710054 China*

**Keywords:** Recommendation System, Reinforcement Learning, Multi-Armed Bandits, Contextual Bandits.

**Abstract:** The paper introduces the widespread application of personalized recommendation systems across various platforms, which provide tailored online services to users based on their historical movements and personal profiles, thereby satisfying individualized needs and enhancing user experience and satisfaction. It identifies the core objectives of recommendation systems as recommending items most likely to bring maximum utility to users while exploring users' potential interest points to balance the "Exploration vs. Exploitation" problem. The paper proposes a recommendation system based on the Multi-Armed Bandit (MAB) model, which combines the Upper Confidence Bound (UCB) algorithm and the Linear Upper Confidence Bound (LinUCB) algorithm into the mixed algorithm, incorporating context-aware selection, dynamic adjustment, and weighted averaging to optimize the effectiveness of the hybrid algorithm, ensuring robust and reliable decision-making in diverse scenarios. It dynamically adjusts weights to balance the recommendation needs for new and existing users, thereby improving overall system performance and user satisfaction. The system design addresses the challenges of limited interaction records for new users and limited interaction information and user features within the system.

## 1 INTRODUCTION

The widespread application of personalized recommendation systems across various platforms has provided users with tailored online services (Linden et al., 2003), greatly satisfying their individualized needs and effectively enhancing user satisfaction. In each interaction where users seek new items, the system employs personalized recommendations based on users' historical movements and personal profiles. Among these, collaborative filtering (Hill et al., 1995), and content-based recommendation (Resnick and Varian, 1997) are common and classical methods.


The core objectives of recommendation systems can be summarized into two aspects: firstly, recommending items that are most likely to bring maximum utility to users, ensuring they receive recommendations aligned with their interests; secondly, exploring users' potential interest points by attempting to recommend new items that users have not yet discovered but may find interesting. The balance between these two objectives constitutes the

"Exploration vs. Exploitation" (EvE) problem (Bubeck and Cesa-Bianchi, 2012).

"Exploitation" refers to recommending items that the system already knows users will like, focusing on maximizing short-term gains; whereas "Exploration" entails trying out recommendations that the system is less certain whether users will like, in order to gather more information for making better recommendations in the future, emphasizing the maximization of long-term gains. The core challenge of the EvE problem lies in how to balance these two strategies to simultaneously meet users' current needs and continually improve the accuracy and effectiveness of the recommendation system (Gangan et al., 2021).

The Multi-Armed Bandit (MAB) model provides an intuitive and effective framework for addressing the "Exploration vs. Exploitation" problem in recommendation systems. It simplifies the recommendation problem into a gambling scenario where pulling each "arm" represents a recommendation action, aiming to maximize total rewards. However, the standard MAB model

---

<sup>a</sup> <https://orcid.org/0009-0005-4875-3946>

primarily focuses on balancing between different choices without directly utilizing users' historical records and feature information. This limitation may result in insufficient recommendation accuracy, as the model fails to fully leverage available personalized user information to optimize its decision-making process (Bubeck and Cesa-Bianchi, 2012).

Contextual Bandits represent a significant improvement over the standard MAB model by incorporating additional context information to enhance the performance of recommendation systems. These context cues can include users' historical movements, preferences, demographic features, etc. This enhancement allows recommendation systems to better consider users' individualized needs, thereby achieving more precise and personalized recommendations. By incorporating context information, the model can better predict user responses, leading to more relevant recommendations and higher user satisfaction. Thus, Contextual Bandits offer an effective solution to address the Exploration vs. Exploitation problem in recommendation systems.

The recommended system proposed in this paper addresses the challenges of limited interaction records for new users and limited overall interaction information and user features within the system. It adopts a strategy combining the Upper Confidence Bound (UCB) algorithm and the Linear Upper Confidence Bound (LinUCB) algorithm. Different strategies are employed in system design to address different environments. In the movie background experiment, accumulated regret and accumulated reward are used as criteria to evaluate the superiority of algorithms. By adjusting the strategy, the goal is to maximize accumulated reward to find the algorithm that yields the highest user ratings, thereby enhancing the overall performance of the system and user satisfaction.

## 2 SYSTEM MODEL AND ALGORITHMS

In this section, the MAB model and the algorithms utilized in the experiments will be discussed, with a particular emphasis on effectively balancing exploration and exploitation in a dynamically changing environment.

### 2.1 Contextual Bandits

In the framework of contextual bandits (Chu et al., 2011), consideration is given to a scenario of a recommendation system. Assume there is a group of

users  $i = 1, 2, \dots, m$ , where each user has a feature vector  $a_i \in \mathbb{R}^p$ , and a set of items  $j = 1, 2, \dots, n$ , where each item has a feature vector  $b_j \in \mathbb{R}^p$ . These feature vectors are combined into matrices  $A \in \mathbb{R}^{p \times m}$  and  $B \in \mathbb{R}^{p \times n}$  representing the feature information of users and items, respectively.

At each time step  $t \in \mathbb{Z}^+$ , a user  $i(t)$  enters the system and is recommended an item  $j(t)$ , then provides a rating  $r(t)$ , where the rating is determined by the dot product of the user's feature vector and the item's feature vector, along with an error term  $z(t)$ . Specifically, the rating is computed as follows:

$$r(t) = a_{i(t)}^T b_{j(t)} + z(t) \quad (1)$$

The goal is to design a recommendation approach that maximizes the cumulative reward after multiple recommendations, or equivalently, minimizes the cumulative regret, which is defined as the accumulation of differences between the true reward and the predicted reward in each round. Specifically, the regret  $\Delta(t)$  in each round can be expressed as:

$$\Delta(t) = r(t) - a_{i(t)}^T b_{j(t)} \quad (2)$$

Then, the cumulative regret  $R_T$  can be obtained by summing up the regrets in each round:

$$R_T = \sum_{t=1}^T \Delta(t) \quad (3)$$

By computing the cumulative regret, the system can assess the performance of the recommendation algorithm. Within the framework of contextual multi-armed bandits, a strategy can be devised to minimize cumulative regret by dynamically selecting recommended items.

### 2.2 UCB Algorithm

Table 1: UCB Algorithm pseudocode.

Algorithm 1 UCB Algorithm	
1.	Initialize: $n = 0$ , $Q(a) = 0$ , $N(a) = 0$ for all actions $a$
2.	<b>for</b> each experiment <b>do</b>
3.	<b>for</b> $t = 1, 2, \dots, T$ <b>do</b>
4.	<b>for</b> each user $i$ <b>do</b>
5.	Choose <span style="float: right;">action</span>
	$a_t = \text{argmax}_a(UCB_a)$
6.	Take action $a_t$ , observe reward $r_t$
7.	Update: $n = n + 1$
8.	$N(a_t) = N(a_t) + 1$
9.	$Q(a_t) = Q(a_t) + 1$
10.	<b>end for</b>
11.	Update cumulative regret
12.	<b>end for</b>
13.	<b>end for</b>

UCB algorithm is a widely used approach for

addressing the multi-armed bandit problem, which finds applications across various domains. Its core principle revolves around striking an optimal balance between exploration and exploitation (Lattimore and Szepesvári, 2020). By computing the UCB index for each arm, it determines the next action to maximize cumulative rewards. The key lies in selecting an appropriate exploration parameter  $c$  (or  $\beta$ ) and employing a confidence interval calculation formula to update the UCB indexes of arms, effectively balancing exploration of unknown arms and exploitation of known arms.

Denote the empirical mean reward of arm  $j$  up to time  $t$  as  $\mu_j(t)$ , and the number of times arm  $j$  has been selected up to time  $t$  as  $n_j(t)$ . Then, the UCB index  $UCB(j, t)$  for arm  $j$  at time  $t$  can be defined as:

$$UCB(j, t) = \mu_j(t) + c \sqrt{\frac{\log t}{n_j(t)}} \quad (4)$$

Here,  $c$  is a tuning parameter that controls the level of exploration. It is typically chosen to balance exploration and exploitation, with a common choice being  $c = 2$ .

To derive the UCB algorithm, it is necessary to determine how to select the arm with the highest UCB at each time step  $t$ , and then update the empirical mean reward and the number of selections based on the observed reward. The arm with the highest UCB index at time  $t$  is selected as:

$$j(t) = \operatorname{argmax}_j UCB(j, t) \quad (5)$$

After selecting arm  $j(t)$  and observing the reward  $r(t)$ , the empirical mean reward and the numbers of selections are updated as follows:

$$\mu_{j(t)}(t + 1) = \frac{n_{j(t)}(t) * \mu_{j(t)}(t) + r(t)}{n_{j(t)}(t) + 1} \quad (6)$$

$$n_{j(t)}(t + 1) = n_{j(t)}(t) + 1 \quad (7)$$

This process is repeated for each time step, allowing the UCB algorithm to dynamically adjust its arm selection strategy based on the observed rewards and the exploration-exploitation trade-off.

### 2.3 LinUCB Algorithm

Table 2: LinUCB Algorithm pseudocode.

Algorithm 2 LinUCB Algorithm	
1.	Initialize: User models $\{A_i, b_i\}$ for each user $i$ , where $A_i$ is the identity matrix and $b_i$ is a vector of zeros
2.	<b>for</b> each experiment <b>do</b>
3.	<b>for</b> $t = 1, 2, \dots, T$ <b>do</b>

4.	<b>for</b> each user $i$ <b>do</b>
5.	Calculate LinUCB values $UCB_a$ for each arm $a$ based on user model $\{A_i, b_i\}$ and exploration parameter $\alpha$
6.	Choose action $a_t^i = \operatorname{argmax}_a UCB_a$
7.	Simulate pulling arm $a_t^i = \operatorname{argmax}_a UCB_a$
8.	Update user model $\{A_i, b_i\}$ based on observed reward $r_t^i$
9.	<b>end for</b>
10.	Update cumulative regret
11.	<b>end for</b>
12.	<b>end for</b>

LinUCB algorithm extends the UCB framework by incorporating contextual information about arms, making it suitable for more complex scenarios where arm performance is dependent on contextual features. It operates on the principle of linear contextual bandits, where the reward prediction is modeled as a linear combination of the arm's features and learned coefficients (Wang et al., 2022). The decision-making process involves selecting the arm with the highest predicted reward, adjusted by an exploration factor that accounts for the uncertainty in the estimation. Key components include a feature vector for each arm, a dynamically updated parameter vector, and an exploration parameter  $c$  (or  $\beta$ ), which influences the width of the confidence interval used in selecting arms, thereby balancing the trade-off between exploring less certain options and exploiting known rewards.

In the LinUCB algorithm, the relationship between arms (items) and their features is modeled using linear regression. This model calculates the expected reward of an arm  $j$  at time  $t$ , denoted  $\hat{\mu}_j(t)$ , as the inner product of the regression coefficient vector  $\theta_j(t)$  and the feature vector  $a_{i(t)}$  of the user selecting arm  $j$ :

$$\hat{\mu}_j(t) = \theta_j(t)^T a_{i(t)}$$

The LinUCB algorithm seeks to maximize the UCB index for each arm at each time step  $t$ . The UCB index for arm  $j$  at time  $t$ , is defined as:

$$UCB(j, t) = \theta_j(t)^T a_{i(t)} + c \sqrt{a_{i(t)}^T A_j(t)^{-1} a_{i(t)}}$$

Here,  $A_j(t)$  represents the covariance matrix of arm  $j$  up to time  $t$ , and  $c$  is a tuning parameter that controls the trade-off between exploration and exploitation. The term

$$\sqrt{a_{i(t)}^T A_j(t)^{-1} a_{i(t)}}$$

quantifies the uncertainty in the estimated mean reward.

The arm selected at each time step  $t$  is the one with the highest UCB:

$$j(t) = \operatorname{argmax}_j \text{UCB}(j, t)$$

Upon the selection of arm  $j(t)$  and the observation of the reward  $r(t)$ , updates are made to the regression coefficient vector  $\theta_{j(t)}(t)$  and the covariance matrix  $A_{j(t)}(t)$  based on the observed features and reward. These updates enable the LinUCB algorithm to adaptively modify its arm selection strategy over time, maintaining an optimal balance between exploration and exploitation.

## 2.4 Mixed Algorithm

In this part, the application of different strategies to adjust the two algorithms will be introduced, aiming to optimize the effectiveness of the system.

### 2.4.1 Context-Aware Selection Approach

The mixed algorithm incorporates context-aware selection to determine the appropriate algorithm for each decision point. It begins by assessing whether relevant contextual information is available for the current decision. If contextual information is present and deemed highly correlated with the decision, the algorithm favours the LinUCB approach. This preference stems from LinUCB's ability to leverage contextual information, such as user features and historical movements, to more accurately estimate the potential rewards for each action.

### 2.4.2 Dynamic Adjustment Approach

To ensure flexibility and responsiveness, the mixed algorithm employs a dynamic adjustment strategy. Periodically, after a fixed number of time steps, denoted as 'n,' the algorithm evaluates the performance of both the UCB and LinUCB algorithms. Based on this comparison, it dynamically selects the algorithm that demonstrates superior performance over the recent history. This dynamic adjustment mechanism allows the algorithm to adapt to changing conditions and evolving patterns in the data.

### 2.4.3 Weighted Averaging Approach

In addition to context-aware selection and dynamic adjustment, the mixed algorithm incorporates weighted averaging to combine the outputs of the UCB and LinUCB algorithms. Each algorithm is

assigned an index weight of 0.5, reflecting an equal contribution to the final decision-making process. By averaging the results of both algorithms, the mixed approach aims to mitigate the individual weaknesses of each algorithm while leveraging their respective strengths. This balanced integration ensures robust and reliable decision-making in diverse scenarios.

## 3 IMPLEMENTATION AND EVALUATION

For this experiment, the dataset employed is the MovieLens dataset (Harper and Konstan, 2015). The dataset contains information on movies, users, and user ratings, with a total of 18 movie tags. Each time the arm is pulled, the system selects a tag and returns a movie rating containing that tag as a reward. The regret is defined as the difference between the optimal possible reward and the gained reward. Through experiments, the aim is to find the best-performing strategy that maximizes accumulated rewards and minimizes accumulated regrets.

Considering the real-time feedback characteristic of online recommendation systems (Zhang et al., 2020), the experiment step size  $T$  is set to 10,000, and the number of runs  $N$  for each algorithm is set to 10 to avoid randomness.

For each output of experimental results, cumulative regrets and cumulative rewards, after  $N$  trials, the final result for mean cumulative regrets is calculated as:

$$\text{mean\_regrets} = \frac{1}{N} \sum_{i=1}^N \text{cumulative\_regrets}$$

$$\text{mean\_rewards} = \frac{1}{N} \sum_{i=1}^N \text{cumulative\_rewards}$$

The experiment sets the optimal possible reward for each time step  $t$  to be 5.

Figure 1 displays the cumulative regrets and cumulative rewards of the UCB algorithm based on four exploration coefficients  $\beta$  under the MovieLens dataset. It can be observed that the smallest exploration coefficient  $\beta$  yields the best algorithm performance. This indicates that using the UCB algorithm on this dataset makes it easy to identify the arm with the highest average reward without the need for excessive exploration of all arms, thus reducing unnecessary resource wastage.

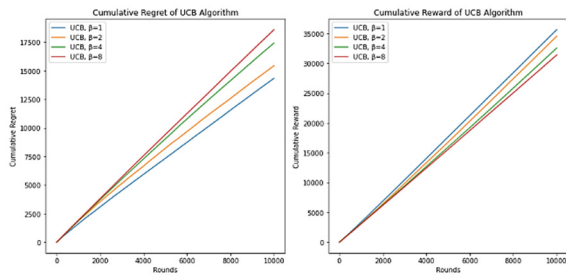


Figure 1: UCB Algorithm results.

Figure 2 displays the cumulative regrets and cumulative rewards based on four exploration coefficients  $\beta$  in the LinUCB algorithm under the given dataset. It can be observed that the performance is poorest when  $\beta = 2$ , while  $\beta = 1$  and  $\beta = 4$  show relatively better performance. With the given number of steps  $T$ , an average reward of above 3.5 can be achieved.

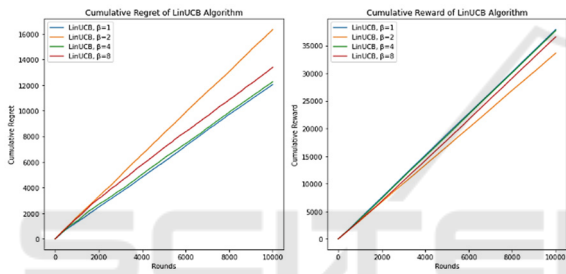


Figure 2: LinUCB Algorithm results.

Figure 3 illustrates the cumulative regrets and cumulative rewards of the mixed algorithm mentioned earlier, including the results of the UCB algorithm to aid in comparing algorithm performance. It can be observed that the Weighted Averaging approach performs significantly worse than other algorithms, indicating that the allocation of exponential weights has a significant impact on algorithm results. The Dynamic Adjustment Approach is the optimal algorithm in this environment, achieving an average reward of approximately 4 per round by adopting better results every 1000 steps, making it the highest average reward.

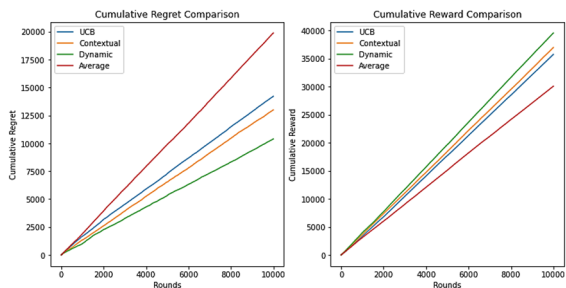


Figure 3: Multi approaches comparison.

## 4 CONCLUSIONS

In summary, this paper proposes a novel recommendation system based on the Multi-Armed Bandit model, integrating the UCB and LinUCB algorithms. The system dynamically balances exploration and exploitation to effectively optimize user satisfaction and system performance, thus addressing the challenges of personalized recommendation.

Through empirical evaluations, the effectiveness and feasibility of the hybrid algorithm in various scenarios have been demonstrated. However, there are still research deficiencies, such as the inability of the context-aware hybrid algorithm to effectively demonstrate its advantages in environments where the data is complete. Future research could explore more advanced techniques, such as integrating additional contextual information or enhancing the system's adaptive learning capabilities. Additionally, studying real-world applications and user research will provide valuable insights into the practical utility and user acceptance of the recommendation system.

## REFERENCES

Linden, G., Smith, B., & York, J. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 76-80.

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. 1995. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '95)*. ACM Press/Addison-Wesley Publishing Co., 194-201.

Resnick, P., Varian, H. R. 1997. Recommender systems. *Communications of the ACM*, 40(3), 56-58.

Bubeck, S., Cesa-Bianchi, N. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1), 1-122.

Gangan, E., Kudus, M., & Ilyushin, E. 2021. Survey of multiarmed bandit algorithms applied to recommendation systems. *International Journal of Open Information Technologies*, 9(4), 12-27.

Chu, W., Li, L., Reyzin, L., & Schapire, R. 2011. Contextual Bandits with Linear Payoff Functions. *Proceedings of the Fourteenth International Conf. on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research*, 15, 208-214.

Lattimore, T., Szepesvári, C. 2020. Bandit algorithms. *Cambridge University Press*.

Wang, D., Men, C., & Wang, W. 2022. A contextual multi-armed bandit recommendation algorithm for nuclear power. *Journal of Intelligent Systems*, 17(03), 625-633.

- Harper, F. M., & Konstan, J. A. 2015. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4), Article 19.
- J. Zhang, Y. Wang, Z. Yuan, & Q. Jin, April 2020, "Personalized real-time movie recommendation system: Practical prototype and evaluation," in *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 180-191.

