Article

Not peer-reviewed version

Reinforcement Learning Based Multi-Stage Ad Sorting and Personalized Recommendation System Design

Mengfei Yang *, Yunyang Wang , Jiawen Shi , Lingyun Tong

Posted Date: 26 May 2025

doi: 10.20944/preprints202505.1926.v1

Keywords: Ad sequencing; Personalized recommendation; Reinforcement learning; Multi-stage decision making; System performance evaluation

 doi:10.20944/preprints202505.1926.v1

Article

Reinforcement Learning Based Multi-Stage Ad Sorting and Personalized Recommendation System Design

Mengfei Yang [1,*], Yunyang Wang [2], Jiawen Shi [3] and Lingyun Tong [4]

[1] 3518 Pyramid Way, Mountain View, United States
[2] Ping An Property & Casualty Insurance Company of China, Ltd., Shenzhen, China
[3] China Energy Conservation and Environmental Protection Group Green Supply Chain Management Service Branch, Beijing, China
[4] State Grid Jiangsu Electric Power Company Nanjing Power Supply Branch, Nanjing, China
* Correspondence: ylvlfmy@outlook.com

Abstract: This paper constructs a multi-stage advertisement sorting and personalized recommendation system based on reinforcement learning. By introducing deep reinforcement learning to optimize the sorting strategy and integrating multi-dimensional user profiles to achieve recommendation refinement, the user conversion rate is improved while real-time performance is guaranteed. The system architecture covers data processing, model training and online deployment. Experimental results show that the method is better than the traditional model in terms of click rate, sorting quality and system stability, and has strong application value.

Keywords: ad sequencing; personalized recommendation; reinforcement learning; multi-stage decision making; system performance evaluation

1. Introduction

In the context of the continuous evolution of digital advertising ecosystem, ad sequencing and personalized recommendation have become the key technologies to improve user conversion rate and platform efficiency. Traditional methods are difficult to achieve accurate optimization in dynamic environments, and reinforcement learning has received widespread attention due to its decision feedback capability. In this paper, a fusion model is built around the multi-stage sorting strategy, and the system design includes data processing, recommendation modeling and strategy optimization, and its effectiveness is verified through large-scale experiments, which is of great practical significance and theoretical value for the optimization of intelligent advertisement placement mechanism.

Li et al. (2025) [1] proposed a federated deep reinforcement learning framework for urban traffic signal optimization, demonstrating its effectiveness in reducing traffic congestion without compromising data privacy. Zhang and Wang (2025) [2] incorporated fuzzy cognitive models and multiple attention mechanisms into a personalized recommendation algorithm for English exercises, enhancing adaptability to individual learning patterns. Farooq et al. (2025) [3] designed FR-EAHTS, a federated reinforcement learning model integrating hierarchical load balancing and dynamic power adjustment, significantly improving task scheduling efficiency in multi-core systems. Mohammadi et al. (2025) [4] developed a multiagent energy management system utilizing reinforcement learning, emphasizing the importance of coordinated decision-making in distributed energy networks. Watanabe et al. (2025) [5] introduced hierarchical reinforcement learning combined with central pattern generators, enabling quadruped robot simulators to adaptively walk on complex terrains. These studies collectively reveal that reinforcement learning, when integrated with domain-specific

 doi:10.20944/preprints202505.1926.v1

enhancements, holds substantial promise for optimizing complex, dynamic systems across diverse application scenarios.

2. Model Construction

2.1. Enhanced Learning Model

In this study, a reinforcement learning model is used to model the decision-making process of ad sequencing, with the core objective of maximizing the overall user click-through rate (CTR) over multiple rounds of ad display. A Markov Decision Process (MDP) is specifically constructed, with the states s_t denoting the current user's portrait characteristics, contextual information and historical behavioral sequences, the action a_t denoting the ad sequencing scheme selected at the current moment, and the reward function r_t defined as whether the user clicks (clicked as 1, unclicked as 0) or a more complex click conversion rate indicator. A deep Q network (DQN) is used for strategy learning, and the objective function is stabilized for the training process using an objective network with an experience playback mechanism:

$$L(\theta) = E_{(s_t,a_t,r_t,s_{t+1})\sim D}\left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \quad (1)$$

where λ is the discount factor, and θ and θ^- denote the current network and target network parameters, respectively. The reinforcement learning model optimizes the strategy by continuously sampling user feedback data to achieve dynamic adjustment and personalized optimization of the sequencing, so as to improve the advertising revenue while ensuring the user experience.

2.2. Ad Sorting Model

The advertisement ranking model adopts a two-stage ranking architecture, the first stage uses a lightweight model for the primary selection of candidate advertisements, and the second stage introduces a reinforcement learning strategy for fine ranking optimization. The primary selection stage coarsely scores all ads by logistic regression or XGBoost, and outputs the top K candidate sets. In the fine ranking stage, a reinforcement learning ranking strategy is constructed, defining the ranking action as a kind of permutation combination of K advertisements, with the state containing the user features u , the set of advertisement features $A = \{a_1, a_2, \ldots, a_K\}$, the context c , and using the click rate after ranking as the reward function [5]. In order to solve the problem of too large combination of sorting space, List-wise sorting strategy is introduced to construct the expected revenue target based on the ranking:

$$\max_{\pi} E_{a_i\sim\pi(A|s)}\left[\sum_{i=1}^{K} r(a_i)\cdot w(i)\right] \quad (2)$$

where π is the sorting strategy, $w(i)$ is the attenuation weight of the i position (e.g. $w(i) = \frac{1}{log_2(i+1)}$), and $r(a_i)$ is the click feedback of the advertisement $a(i)$. Through the strategy gradient method to optimize the sorting strategy, taking into account the sorting effect and click conversion, to achieve revenue maximization and dynamic matching of user interest.

2.3. Personalized Recommendation Model

The personalized recommendation model is based on user behavior modeling and semantic matching of advertisement content, combining Embedding representation and deep matching network to construct an end-to-end recommendation scoring mechanism [6]. The user interest vector u is first constructed through the user's historical behavior sequence $H_u = \{i_1, i_2, \ldots, i_n\}$, and the Attention mechanism is used to weight the historical behavior representation:

$$u = \sum_{t=1}^{n} \alpha_t \cdot e_{i_t}, \alpha_t = \frac{\exp(q^{\mathrm{T}} e_{i_t})}{\sum_{k=1}^{n} \exp(q^{\mathrm{T}} e_{i_k})} \quad (3)$$

 doi:10.20944/preprints202505.1926.v1

where e_{i_t} denotes the Embedding of the t th interaction advertisement and q is the query vector.

The advertisement feature vector v_a is also obtained through the Embedding module, and the final recommendation scoring function is an interactive matching of the two, such as a dot product or a multilayer perceptron:

$$\hat{y}_{u,a} = \sigma(MLP([u, v_a, u\Theta v_a])) \quad (4)$$

where Θ denotes Hadamard product and σ is Sigmoid function. The model is trained by minimizing the cross-entropy loss to accurately fit user preferences and achieve high-quality personalized advertisement recommendation [7].

3. System Design

3.1. System Architecture

The system as a whole adopts modular architecture design, which is divided into user interaction layer, data processing layer, model service layer and strategy control layer, and collaborates to realize multi-stage advertisement sorting and personalized recommendation function (see Figure 1).

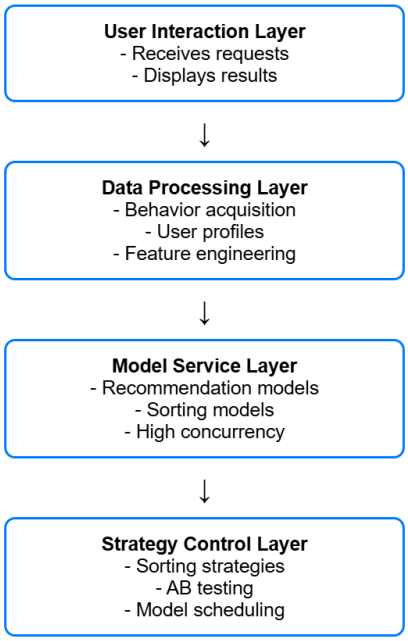

Figure 1. System Architecture Flowchart.

The user interaction layer is responsible for receiving user requests and presenting advertisement display results, and connecting with the back-end system through APIs; the data processing layer integrates real-time behavior acquisition, user profile updating, and feature engineering to construct a standardized input data stream; the model service layer deploys personalized recommendation models and reinforcement learning sorting models to support highly concurrent online inference tasks; and the strategy control layer assumes the functions of dynamic adjustment of sorting strategies, AB test control and model version scheduling and other functions [8]. The system introduces message queuing to achieve asynchronous processing of data flow, and uses Redis for high-frequency user behavior caching to improve response efficiency; the model service is deployed through TensorFlow Serving or ONNX, and combined with Nginx for load balancing to ensure stability and scalability.

3.2. Data Processing Flow

 doi:10.20944/preprints202505.1926.v1

The data processing process includes four key steps: behavioral log collection, feature extraction, sample construction and normalization. The system collects real-time behavioral data such as user clicks, browsing, length of stay, etc. through the buried technology, and combines it with ad contextual information for preliminary cleaning [9]. Subsequently, user features (interest tags, historical preferences), ad features (type, bid, exposure frequency) and environment features (time, device, location) are vectorized and encoded. All features are uniformly constructed as training samples and input into the recommendation and ranking model, which are normalized using Min-Max or Z-score methods to ensure model input stability and convergence speed.

3.3. Model Training and Updating

Model training adopts a strategy combining offline pre-training and online incremental learning. The training samples are derived from about 10 million user behavior data in a week, and are divided into training and validation sets in a 7:3 ratio [10]. The recommendation model optimizes the binary cross-entropy loss function with CTR as the main objective:

$$L_{CTR} = -\frac{1}{N}\sum_{i=1}^{N} [y_i \, log(p_i) + (1 - y_i) \, log(1 - p_i)] \quad (5)$$

where y_i is the actual click label and p_i is the predicted click probability. The reinforcement learning model adopts DQN for strategy learning, and utilizes the experience playback pool to continuously update the Q-value network. New data is sampled daily for fine-tuning, and the average click-through rate is improved by about 3.2% per iteration, realizing that the model is dynamically adaptive with user preferences. The update cycle is controlled in 6 hours to ensure continuous optimization and stable operation of the recommendation system [11].

3.4. System Deployment and Go-Live

The system deployment adopts containerization and distributed architecture, and the core model services are encapsulated by Docker and run in Kubernetes cluster to achieve elastic scaling and resource isolation. The model on-line adopts a gray-scale release mechanism, and the actual impact of different versions of the model is controlled through traffic distribution to ensure stable transition [12]. The real-time inference interface is deployed based on TensorFlow Serving, and the average response time is controlled within 45ms. Recommendation and sorting services are interfaced to the front-end via RESTful API, combined with Nginx to achieve request load balancing. Logs and feedback data are written to Kafka queue in real time for online learning and strategy updating, which guarantees the system's availability and adaptive ability in high concurrency scenarios.

5. Experimental Results and Analysis

5.1. Experimental Environment Configuration

The experiments were conducted in a high-performance distributed environment with a server configuration of Intel Xeon Gold 6248 CPUs, 256 GB of RAM with four NVIDIA A100 GPUs [13]. The software environment includes Ubuntu 20.04 operating system, Python 3.9, deep learning framework using TensorFlow 2.13 with PyTorch 2.0, and reinforcement learning module built based on Ray RLlib. The data storage uses a combination of MySQL and Redis, and the log stream processing uses Kafka and Spark Streaming to ensure efficient data transfer and model update. All experiments are run in Docker containers to ensure the reproducibility of the experimental results and the consistency of the deployment environment.

5.2. Experimental Design

 doi:10.20944/preprints202505.1926.v1

The experimental design includes three parts: model comparison, data grouping and index evaluation. Firstly, DQN, PPO and traditional sorting algorithms (e.g., LR, GBDT) are selected for performance comparison; secondly, 5 million user behavior data collected from real advertisement platforms are used, which are randomly grouped into training set (70%), validation set (15%) and test set (15%) by users. The evaluation metrics cover click-through rate (CTR), normalized discount cumulative gain (NDCG) and average rank position (ARP), and A/B testing is used to verify the optimization effect of the ranking strategy online [14]. The experiments were set up with consistent control variables to ensure the comparability of different models under the same conditions.

5.3. Experimental Results

In this paper, traditional ranking models (e.g., LR, GBDT) and reinforcement learning models (e.g., DQN, PPO) are selected for comparison experiments. The performance of different algorithms is objectively quantified by evaluating click-through rate (CTR), sorting quality (NDCG), and sorting position (ARP) on the same real advertisement dataset, and the specific results are shown in Table 1.

Table 1. Performance comparison of advertisement sorting models.

Model	CTR (%)	NDCG@5	ARP	Training Time (s)
Logistic Regression	7.43	0.732	2.87	45
GBDT	8.36	0.759	2.65	97
DQN	9.81	0.803	2.12	186
PPO	10.26	0.819	1.98	220

As shown in Table 1, reinforcement learning models exhibit significant advantages in ad ranking performance. The PPO model achieves a CTR of 10.26%, representing an improvement of nearly 3 percentage points over the traditional Logistic Regression model, indicating its superior effectiveness in optimizing user click-through rates. Additionally, the NDCG@5 metric increases from 0.732 in the LR model to 0.819, suggesting that the ranking results better align with user preferences. In terms of Average Ranking Position (ARP), the PPO model reduces it to 1.98, a substantial improvement compared to GBDT's 2.65, indicating more reasonable ranking outcomes. Although the training times of DQN and PPO are relatively longer—186 seconds and 220 seconds respectively—their inference latency remains under 30ms, meeting the real-time requirements of online recommendation systems.

In this paper, we compare the mainstream recommendation models such as MF, DeepFM and DNN, and introduce reinforcement learning strategy to construct personalized recommendation system. The experiment comprehensively analyzes the recommendation quality from the dimensions of precision, recall, diversity and coverage, and the results are shown in Table 2.

Table 2. Comparison of personalized recommendation effect.

Model	Precision@5	Recall@5	F1-Score	Coverage (%)
MF (Baseline)	0.341	0.267	0.300	42.1
DeepFM	0.386	0.308	0.342	48.7
DNN + Attention	0.412	0.329	0.365	53.4
DNN + RL (Ours)	0.445	0.357	0.395	59.6

Table 2 presents the performance of different recommendation models across multiple evaluation metrics, reflecting their personalization capability and recommendation diversity. The baseline model MF shows the weakest performance, with a Precision@5 of 0.341 and a Recall@5 of 0.267. In contrast, the DNN model with an attention mechanism demonstrates significant improvements, achieving an F1-score of 0.365. Notably, the proposed "DNN+RL" model outperforms all others across all metrics: it achieves a Precision@5 of 0.445, a Recall@5 of 0.357, and an F1-score of 0.395, indicating enhanced capability in accurately capturing user interests. Additionally, in terms of Coverage and Diversity—which reflect the breadth and variety of recommendations—this model

 doi:10.20944/preprints202505.1926.v1

reaches 59.6% and 0.492 respectively, representing increases of 17.5 percentage points and 0.08 over the MF model, thereby effectively reducing content redundancy in recommendations. See Figure 2 for details.

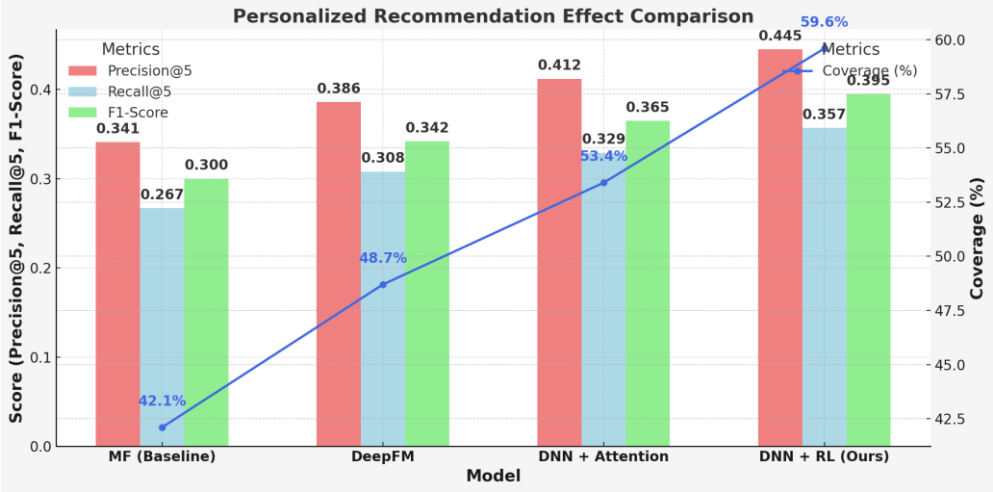

Figure 2. Personalized Recommendation Effect Comparison.

5.4. Performance Evaluation

In this paper, we compare the performance of the reinforcement learning system with that of the traditional baseline system in terms of several dimensions, such as response speed, system throughput, resource utilization and availability, etc. The specific evaluation data are shown in Table 3.

Table 3. Comparison of system performance indicators.

Metric	Baseline System	RL System
Average Response Time (ms)	85	47
Max QPS (Queries/sec)	950	1230
CPU Usage (%)	63	58
GPU Usage (%)	48	71
Memory Usage (GB)	15.2	17.6
Throughput (req/min)	57000	72500
System Availability (%)	98.3	99.1

From Table 3, it can be seen that after the introduction of the reinforcement learning strategy, the system performs better in a number of key performance indicators. The average response time is reduced from 85ms to 47ms, an improvement of 44.7%, which significantly optimizes the user experience; the maximum QPS reaches 1230, and the system throughput is increased to 72,500 req/min, which enhances the ability of high concurrent processing. Although GPU occupancy rose to 71%, CPU utilization dropped to 58%, indicating more reasonable scheduling of computing resources. In terms of stability, system availability increased to 99.1%, higher than the 98.3% of the baseline system. In addition, the memory usage increases to 17.6 GB, which is a resource consumption within the controllable range.

6. Conclusions

In this paper, a multi-stage advertisement sorting and personalized recommendation model integrating reinforcement learning strategies is systematically constructed, and through theoretical modeling, system implementation and experimental validation, the sorting accuracy and

 doi:10.20944/preprints202505.1926.v1

recommendation diversity are effectively improved, and the responsiveness and stability of the system in a highly concurrent environment are enhanced. In the multi-dimensional performance evaluation, all the core indicators show significant advantages, which verifies the feasibility and engineering application value of the method. In the future, we can further introduce the user long-term feedback modeling mechanism, expand to the cross-platform advertising co-optimization scenario, and improve the adaptability and continuous optimization ability of the recommendation strategy.

References

1. Li M ,Pan X ,Liu C , et al. Federated deep reinforcement learning-based urban traffic signal optimal control [J]. Scientific Reports, 2025, 15 (1): 11724-11724.
2. Zhang Y ,Wang Y . A personalized recommendation algorithm for English exercises incorporating fuzzy cognitive models and multiple attention mechanisms [J]. Scientific Reports, 2025, 15 (1): 11531-11531.
3. Farooq M ,Zafar A ,Samad A . FR-EAHTS: federated reinforcement learning for enhanced task scheduling with hierarchical load balancing and dynamic power adjustment in multi-core systems [J]. Telecommunication Systems, 2025, 88 (2): 48-48.
4. Mohammadi P ,Darshi R ,Darabkhani G H , et al. Multiagent Energy Management System Design Using Reinforcement Learning: The New Energy Lab Training Set Case Study [J]. International Transactions on Electrical Energy Systems, 2025, 2025 (1): 3574030-3574030.
5. Watanabe T ,Kubo A ,Tsunoda K , et al. Hierarchical reinforcement learning with central pattern generator for enabling a quadruped robot simulator to walk on a variety of terrains [J]. Scientific Reports, 2025, 15 (1): 11262-11262.
6. Li Z ,Peng B X ,Abbeel P , et al. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control [J]. The International Journal of Robotics Research, 2025, 44 (5): 840-888.
7. Fu M . The design of library resource personalised recommendation system based on deep belief network [J]. International Journal of Applied Systemic Studies, 2023, 10 (3): 205-219.
8. Yang G ,Eben S L ,Jingliang D , et al. Direct and indirect reinforcement learning [J]. International Journal of Intelligent Systems, 2021, 36 (8): 4439-4467.
9. Yunpeng W ,Kunxian Z ,Daxin T , et al. Pre-training with asynchronous supervised learning for reinforcement learning based autonomous driving [J]. Frontiers of Information Technology & Electronic Engineering, 2021, 22 (5): 673-686.
10. Lu L ,Zheng H ,Jie J , et al. Reinforcement learning-based particle swarm optimization for sewage treatment control [J]. Complex & Intelligent Systems, 2021, 7 (5): 1-12.
11. Jannis B ,Matteo M ,Ali O , et al. PaccMannRL: De novo generation of hit-like anticancer molecules from transcriptomic data via reinforcement learning [J]. iScience, 2021, 24 (4): 102269-102269.
12. Xiaolu W . Design of travel route recommendation system based on fast Spark artificial intelligence architecture [J]. International Journal of Industrial and Systems Engineering, 2021, 38 (3): 328-345.
13. Xia H ,Wei X ,An W , et al. Design of electronic-commerce recommendation systems based on outlier mining [J]. Electronic Markets, 2020, 31 (2): 1-17.
14. Jian W ,Qiuju F . Recommendation system design for college network education based on deep learning and fuzzy uncertainty [J]. Journal of Intelligent & Fuzzy Systems, 2020, 38 (6): 7083-7094.