Neural Network-Based Parameter Tuning for Multi-Armed Bandit Algorithms

Yuhan Shi

Department of Computer Science, University College London, London, WC1E6BT, United Kingdom

Abstract. This paper presents a novel approach for dynamically tuning the exploration parameter in Multi-Armed Bandit (MAB) algorithms using Deep Q-Networks (DQN), focusing on enhancing performance in static and dynamic environments. Traditional MAB algorithms such as Upper Confidence Bound (UCB) and Thompson Sampling (TS) rely on fixed exploration parameters and assume stationary reward distributions, limiting their effectiveness in real-world applications where reward distributions can be dynamic. This paper proposes a learning-based method where a DQN agent observes the state of the MAB environment and selects an appropriate exploration parameter from a predefined set to address this problem. Experimental results show that the DQN-enhanced UCB algorithm consistently outperforms its traditional counterpart in both static and dynamic environments by achieving lower cumulative regret. In contrast, DQN-tuned TS moderately improves dynamic settings but exhibits instability in static environments. These findings highlight the potential of integrating neural network-based learning with classical decision-making strategies to enable adaptive exploration in non-stationary environments, offering valuable insights for recommender systems and other sequential decision-making tasks.

1 Introduction

The MAB problem is a classical sequential decision-making problem that models the trade-off between exploration and exploitation [1]. In the standard MAB setting, an agent faces multiple options, often called "arms," each providing stochastic rewards drawn from an unknown distribution. The agent's objective is to maximize the cumulative reward over time by balancing the exploration of uncertain arms and the exploitation of arms believed to yield higher rewards. Representative algorithms addressing the exploration-exploitation dilemma include the UCB and TS. The UCB algorithm selects arms according to an upper bound of the estimated reward, encouraging exploration of less-sampled arms while favouring arms with higher empirical rewards. On the other hand, TS adopts a Bayesian perspective, sampling and updating the reward distribution over each arm to guide the arm selection. UCB and TS have been shown to achieve sub-linear regret bounds and perform well in stationary environments [1].

zcabysh@ucl.ac.uk

However, these classical methods assume that the reward distribution of each arm remains stable over time, which does not hold in the context of non-stationary bandits, where the reward distribution can change dynamically [2]. This study proposes a method that leverages DQN to dynamically adjust the exploration parameters of MAB algorithms. Specifically, this paper aims to enhance the flexibility of classical methods by introducing a learning-based parameter-tuning mechanism that adapts to different environmental conditions.

This paper applies DQN to tune the exploration parameters of UCB and TS during their execution, enabling them to adjust exploration levels according to environmental feedback. The MovieLens dataset is an experimental environment to simulate user feedback for recommendation tasks. The proposed method is evaluated by comparing the cumulative regret of standard UCB and TS algorithms with their DQN-enhanced counterparts. Each algorithm is run to obtain the mean and standard deviation of cumulative regret for comparison. Furthermore, this paper conducts parameter adjustment analysis to observe how the exploration parameter changes during the learning process and an ablation study to verify the effectiveness of DQN tuning on the classic MAB algorithms.

2 Related Work

A significant body of work has focused on improving algorithm performance in the context of non-stationary bandits. One of the classic approaches is window-sliding, where algorithms maintain a fixed-size window of recent observations to update estimates of reward distributions, effectively "forgetting" outdated data. By focusing on the most recent information, the algorithm can adapt to environmental changes [3, 4].

Recent research has extensively explored the application of MAB algorithms in various real-world domains, including recommendation systems, online advertising, and e-commerce [5-7]. Many of these studies recognize the challenge of non-stationary reward distributions, often caused by evolving user preferences. Common approaches focus on incorporating contextual information or leveraging experience to improve the adaptability of MAB algorithms in dynamic environments. One example is hybrid methods that combine traditional MAB algorithms with meta-learning techniques to improve adaptability in non-stationary environments [8]. In this context, meta-learning approaches can learn to predict or initialize optimal hyperparameters for MAB algorithms, such as exploration rates or learning rates, based on patterns observed in past environments. By doing so, the MAB algorithm is better equipped to adjust its behaviour in response to environmental changes, such as evolving user preferences or shifting reward distributions, which are common in real-world applications.

Meanwhile, reinforcement learning (RL) has emerged as a robust framework for sequential decision-making tasks, offering the ability to learn complex policies from interactions with dynamic environments [9]. However, in the context of MAB problems, whether traditional or reinforcement learning approaches, most existing work has primarily focused on directly learning arm-selection policies. Relatively little attention has been given to dynamically adjusting the internal parameters of classical MAB algorithms [10]. This presents a research opportunity to combine the strengths of neural network-based learning with established MAB strategies, enabling adaptive parameter control that responds to different learning stages or changes in the environment. In particular, this paper proposes leveraging DQN, an algorithm that uses a neural network to approximate the action-value function and learn optimal decision policies from high-dimensional inputs [11]. By integrating neural network-based learning with classical exploration-exploitation strategies, this paper aims to improve MAB algorithms' adaptability in real-world recommendation settings. The approach was evaluated using the MovieLens dataset, a

widely used benchmark in recommender systems research that contains user ratings for a diverse set of movies collected from an online movie recommendation platform [12].

3 Methodology

The experiments are conducted using the MovieLens 1M dataset, where users are divided into four age groups representing the arms of the MAB algorithm. Given a movie, the algorithms aim to identify which age group prefers it the most. The reward signal is derived from the normalized user ratings in the dataset, simulating a personalized recommendation scenario. The experiment aims to verify the proposed method's effectiveness by comparing its performance with traditional MAB algorithms that use fixed exploration parameters.

3.1 Data Processing

The MovieLens 1M dataset contains user demographic information, movie ratings, and movie details. The user,

rating, and movie information are first loaded and combined into a single dataset through a merging process to prepare the data for the experiment. The user demographic data includes attributes such as age, gender, and occupation, while the rating data provides the user-movie interactions in the form of rating scores ranging from 1 to 5. To standardize the reward signal for the multi-armed bandit algorithms, the original rating values are normalized into the range [0,1], where higher values indicate stronger user preference. Users are then divided into four distinct age groups, each representing an arm of the bandit algorithm. These age groups are defined as Young (<18), Young Adult (18-34), Middle-Aged (35-49), and Older Adult (50+).

Additionally, to ensure the quality and reliability of the experimental results, the dataset is further refined by filtering out unpopular movies that have received fewer than 100 ratings. This step focuses the evaluation on movies with sufficient user feedback, thereby reducing noise and improving the robustness of the learning process.

3.2 State Definition

Including specific components in the state vector is crucial for enabling the DQN to effectively learn and adjust the exploration parameter c in the context of the UCB algorithm. Table 1 illustrates the structure of the state vector, which includes the estimated mean rewards, the normalized number of pulls, the reward variance for each arm, and the exploration constant. Despite the exploration parameter, all elements are lists of four values corresponding to four arms. Each of these elements provides important information about the current state of the MAB environment.

The mean reward estimates offer insight into the expected outcomes of each arm, helping the model identify which arms are likely to yield higher rewards. The normalized pull frequencies reflect how much each arm has been explored relative to others, guiding the model in balancing exploration and exploitation. The variance indicates the uncertainty in the reward estimates, which is crucial for determining whether further exploration is necessary. Finally, the exploration parameter c directly influences the exploration-exploitation trade-off, and including it in the state enables the DQN to learn when and how to adjust this parameter dynamically. By combining these features, the state vector provides a comprehensive representation of the current environment, empowering the DQN to make informed decisions and improve the performance of the MAB algorithm over time.

Table 1 State vector

Component name	estimated mean rewards	normalized number of pulls	reward variance	exploration parameter	all component
Size	4	4	4	1	13

The mean reward estimates offer insight into the expected outcomes of each arm, helping the model identify which arms are likely to yield higher rewards. The normalized pull frequencies reflect how much each arm has been explored relative to others, guiding the model in balancing exploration and exploitation. The variance indicates the uncertainty in the reward estimates, which is crucial for determining whether further exploration is necessary. Finally, the exploration parameter c directly influences the exploration-exploitation trade-off, and including it in the state enables the DQN to learn when and how to adjust this parameter dynamically. By combining these features, the state vector provides a comprehensive representation of the current environment, empowering the DQN to make informed decisions and improve the performance of the MAB algorithm over time.

3.3 Action Space Definition

In this experiment, the DQN agent aims to choose the best c for the formula from the action space consisting of predefined exploration parameters.

$$UCB_i(t) = \hat{u}_i(t) + c \times \sqrt{\frac{log(n)}{T_i(t)}} \tag{1}$$

The formula (1) represents the reward estimation for each arm in the UCB algorithm. In this formula, $\hat{u}_i(t)$ denotes the estimated expected reward of arm i at time t, computed based on the arm's current average reward and the number of times it has been pulled. The term $T_{\mathrm{i}}(\mathrm{t})$ indicates the number of times arm i has been selected up to time t, while $log(n)$ is the natural logarithm of the total number of rounds. The square root of their quotient gives the confidence bound of each arm, estimating the uncertainty and therefore encouraging exploration of less frequently chosen arms. The parameter c serves as the exploration constant that controls the degree of exploration. The classic UCB algorithm has a fixed c at 1, while in this work, c is dynamically adjusted by the DQN agent.

$$\theta_i \sim N\left(\hat{u}_i\,(t), \frac{c}{T_i(t)}\right) \tag{2}$$

The formula (2) represents the core mechanism of the Gaussian TS algorithm. At each round t, a sample is drawn from a normal distribution defined for each arm, where the distribution is estimated by the estimated mean reward and the reward variance. The $\hat{u}_i(t)$ represents the same estimated mean reward for arm i at time t as described in formula (1). The reward variance is the quotient of the exploration parameter c and $T_{\mathrm{i}}(\mathrm{t})$, where $T_{\mathrm{i}}(\mathrm{t})$ denotes the number of times arm i has been selected up to time t. This formulation estimates the reward distribution for each arm, with the variance decreasing as the number of pulls increases, gradually shifting towards exploitation. This paper will compare the performance between dynamic c and that of fixing c at 1.

Table 2 DQN action space

Action space	0.1	0.25	1	2	4

Table 2 lists all possible actions, each corresponding to a specific value of c that represents a different level of exploration. Smaller values of c encourage more exploitation by focusing on arms with higher estimated rewards, while larger values of c promote more exploration by increasing the likelihood of selecting less-visited arms. By selecting an appropriate c value as an action, the DQN agent learns to adjust c dynamically based on the environment state.

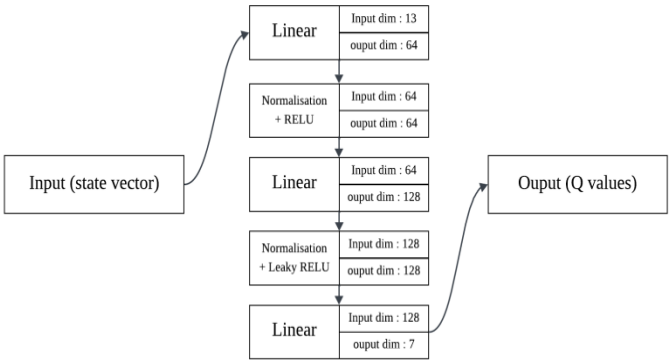

Figure 1 Architecture of the defined DQN network (Photo credit: Original)

Once the state vector and possible actions of the agent are defined, a DQN can be constructed to learn an optimal exploration strategy for the MAB algorithm. The network architecture, shown in Figure 1, takes the 13-dimensional state vector as input. The first layer projects the input into a 64-dimensional space, followed by Layer Normalization and a Rectified Linear Unit (ReLU) activation. The second layer expands the representation to 128 dimensions and applies another Layer Normalization, followed by a Leaky ReLU activation with a negative slope of 0.01. Finally, a linear output layer maps the processed features to Q-values representing discrete candidate values for the exploration parameter.

The network structure is small but effective for tuning parameters in MAB algorithms. Layer Normalization is used in each hidden layer to help keep training stable. This is important due to the non-stationary nature of input distributions, as different arms are explored at varying frequencies. The ReLU activation in the first layer promotes efficient learning by selectively activating relevant features during the early training stages. In the second layer, Leaky ReLU keeps gradients flowing even when the input is negative, which helps the model keep learning in states where the input signal may be weak or less informative.

3.4 Training Parameter

Table 3 summarises the parameters used to train the model. For the two MAB algorithms, the model is trained for 500 epochs. A random movie is selected in each epoch, and the environment is executed for some rounds. The DQN replay function, which is used to sample past experiences from a memory buffer and update model weights, is triggered to train the model every four rounds. This is because the reward is randomly sampled from the reward distribution for each arm. Spacing out the updates gives the model time to collect more diverse reward signals, improving the reliability of the updates.

Table 3 DQN training parameters

Parameter	Description	DQN-UCB Value	DQN-TS Value
Gamma	Discount factor for future rewards	0.95	0.95
Learning rate	Learning rate for updating Q-values	0.001	0.001
Batch size	Size of each training batch	64	128
Replay memory size	Maximum size of the experience replay memory	5,000	10000
Number of rounds	Times of MAB algorithm execution for each epoch	2000	10000

To achieve better performance, the training for DQN-TS requires more rounds, a larger replay memory size, and a larger batch size. This is because the Gaussian Thompson Sampling approach relies on random sampling from the estimated reward distribution, introducing inherent randomness into the decision-making process. The model needs access to a wider range of representative and diverse experiences to learn from this stochastic behaviour effectively.

3.5 Performance Evaluation

The evaluation will compare regrets, analyze parameters, and conduct ablation studies in both static and dynamic environments. In the static environment, the algorithm will be tested on 100 random movies. In the dynamic environment, the algorithm will be applied to one selected movie 100 times, with a sine function added to the reward distribution to simulate changing rewards.

In the cumulative regret experiment, the DQN-tuned MAB algorithms will be compared to the original algorithms that have fixed parameters. Cumulative regret will be measured over time to assess the learning efficiency and performance of each method. The calculation of cumulative regret can be found in Formula (3).

$$R = \sum_{t=1}^{n} (Optimal\ reward\ -\ Sampled\ Reward) \quad (3)$$

In this formula, R denotes the final cumulative regret, and n represents the total number of rounds. The optimal reward is the maximum expected reward from the true reward distribution of each arm, while the sampled reward is the reward sampled from the actual distribution when the algorithm selects a particular arm at time t. A lower cumulative regret indicates that the algorithm is able to identify and exploit the optimal arm faster, which will be the key metric to evaluate the algorithm's performance.

The analysis will also examine how the DQN adjusts its parameters to understand the behavior of the proposed method. The aim is to determine if the DQN agent makes

reasonable adjustments, which will provide insights into how the learned policy adapts to different environments and stages of learning. Additionally, an ablation experiment will verify the role of DQN-based parameter tuning. The performance of the original MAB algorithm with fixed parameters will be compared to the MAB algorithm with randomly adjusted parameters and the proposed MAB algorithm that uses DQN-based tuning. This experiment will demonstrate that any performance improvement is due to intelligent parameter adjustments rather than random fluctuations. Cumulative regret will be the basis for evaluating the effectiveness of the proposed tuning mechanism.

4 Results and Analysis

To evaluate the effectiveness of DQN-tuned MAB algorithms, this paper compared them against the classic MAB algorithms over multiple runs in both static and dynamic environments.

4.1 Static Environment

As shown in Figure 2, the DQN-tuned UCB consistently achieved lower cumulative regret in the static environment, indicating better decision-making and arm selection over time. The mean cumulative regret curve of DQN-UCB rises more slowly than that of UCB, demonstrating its advantage in environments with fixed reward distributions. Figure 3 plots the exploration parameter chosen by the DQN throughout the learning process. The plot reveals a notable pattern: the model typically starts with a higher value of c in the early rounds, favouring exploration. A shift to smaller c values as the algorithm gains confidence in its estimates, effectively shifting toward exploitation.

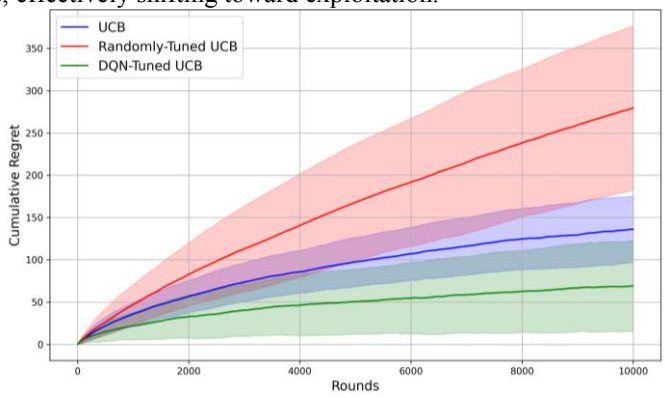

Figure 2 DQN-tuned UCB vs UCB in static environment (Photo credit: Original)

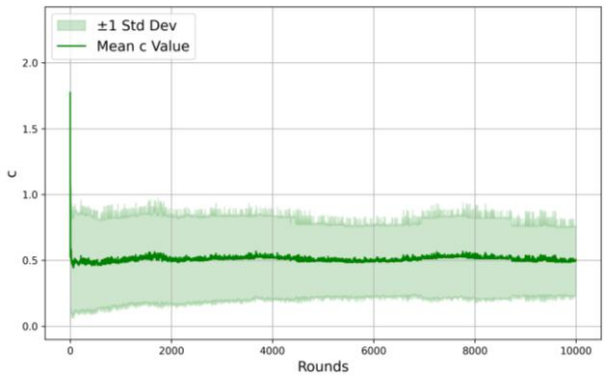

Figure 3 DQN-tuned UCB selection of c in static environment (Photo credit: Original)

Figure 4 shows that DQN-tuned Thompson Sampling has a higher mean and standard deviation in cumulative regret than standard Thompson Sampling. This suggests that although DQN-tuned TS may occasionally outperform standard TS, it tends to be less consistent overall. Figure 5 shows that, unlike the DQN-UCB strategy, the exploration parameter in DQN-tuned TS often begins around 1.0 and then increases to fluctuate around a higher value. This behaviour leads to slower convergence, limiting its advantage over the standard approach in static environments.

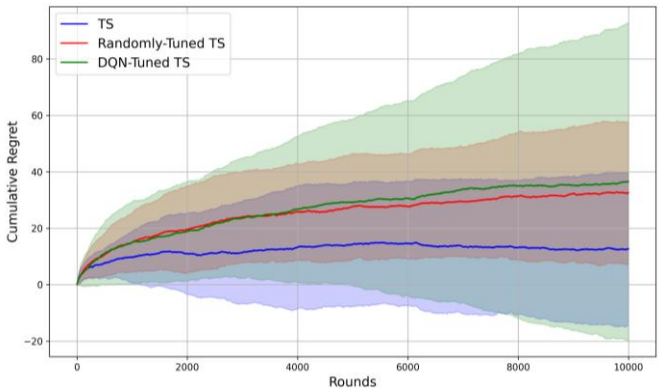

Figure 4 DQN-tuned TS vs TS in static environment (Photo credit: Original)

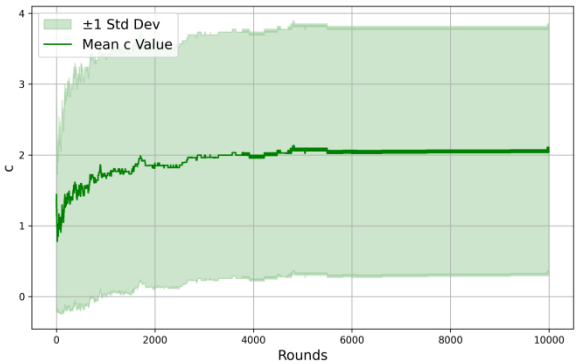

Figure 5 DQN-tuned TS selection of c in static environment (Photo credit: Original)

4.2 Dynamic Environment

In contrast to the static case, the dynamic environment introduces periodic fluctuations in the underlying reward distributions of each arm. Figure 6 illustrates how the mean reward of each age group evolves, with the Middle-aged group initially providing the highest reward. As the environment evolves, the Young group eventually becomes the optimal arm, demonstrating a shift in the reward landscape that challenges fixed exploration strategies.

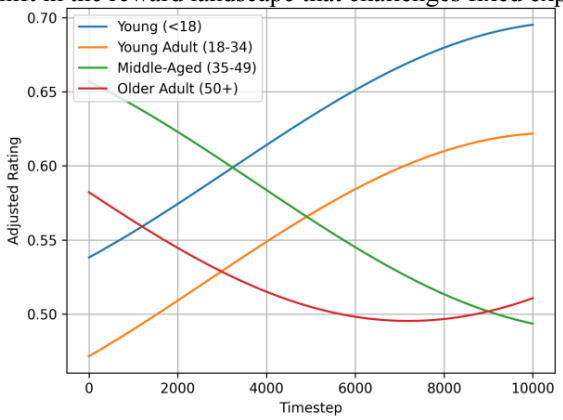

Figure 6 Dynamic reward distribution settings for experiment (Photo credit: Original)

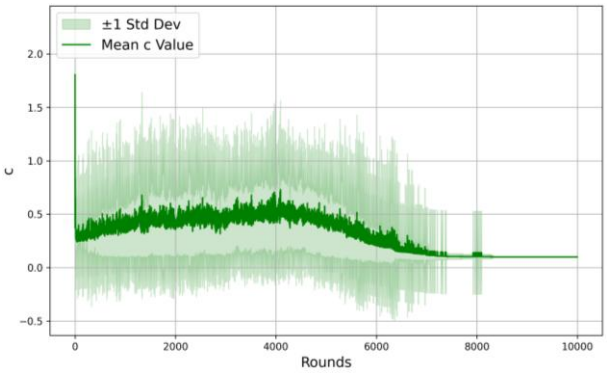

Figure 7 DQN-tuned UCB vs UCB in dynamic environment (Photo credit: Original)

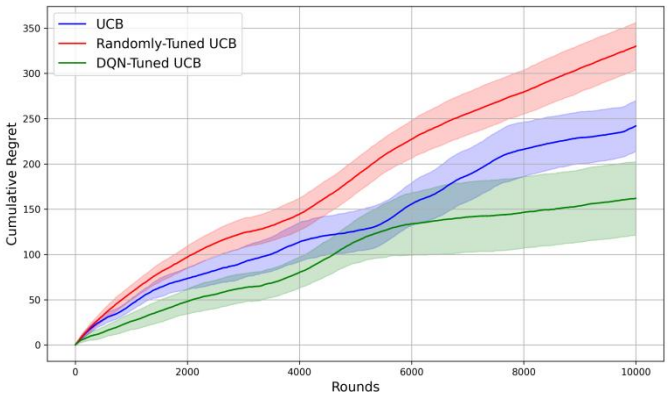

Figure 8 DQN-tuned UCB selection of c in a dynamic environment (Photo credit: Original)

DQN-tuned UCB consistently outperforms the standard UCB algorithm in the dynamic environment, as illustrated in Figure 7. Its cumulative regret curve remains below that of regular UCB throughout the experiment, with the final regret approximately 33% lower. This demonstrates DQN-UCB's ability to adapt more effectively to changes in the reward distribution. Figure 8 shows that, similar to its behaviour in the static environment, the DQN initially selects high exploration values, which gradually decrease as the model gains confidence. However, in response to non-stationary shifts, the DQN increases exploration when it detects environmental changes. This dynamic adjustment is essential in evolving settings and highlights the advantage of learned, adaptive strategies over fixed exploration schedules.

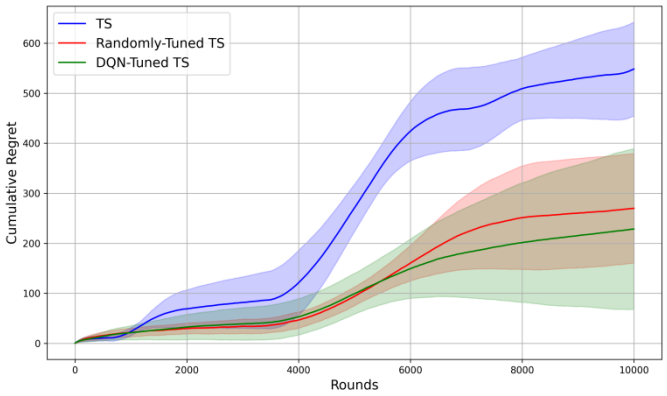

Figure 9 DQN-tuned TS vs TS in dynamic environment (Photo credit: Original)

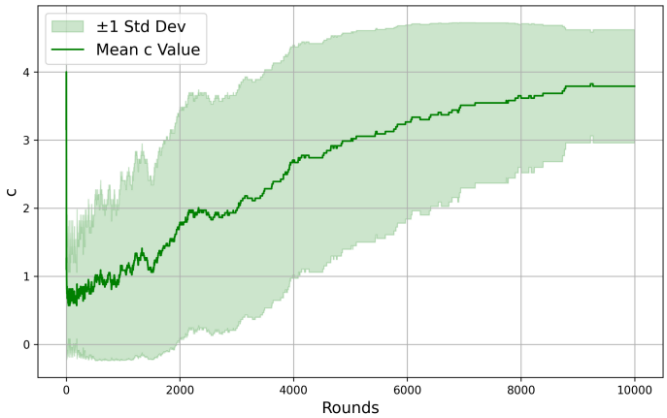

Figure 10 DQN-tuned TS selection of c in a dynamic environment (Photo credit: Original)

As shown in Figure 9, DQN-tuned Thompson Sampling significantly outperforms the standard Thompson Sampling approach in the dynamic environment despite its weaker performance in the static setting. The final mean cumulative regret is approximately 50% lower, highlighting the advantage of adaptive parameter tuning in non-stationary contexts. Figure 10 illustrates how the exploration parameter evolves. DQN-TS adopts a strategy similar to DQN-UCB, where exploration is increased upon detecting changes in the reward distribution.

4.3 Ablation Experiment

The ablation study compares random parameter tuning with standard MAB algorithms and DQN-based tuning. Figures 2 and 7 show that in both static and dynamic environments, the randomly tuned UCB performs the worst, highlighting that DQN-tuned UCB excels due to intelligent parameter selection rather than random exploration.

The results for DQN-TS are mixed. In the static environment (Figures 4 and 9), randomly tuned TS and DQN-tuned TS yield similar mean cumulative regret, but DQN-tuned TS has a larger standard deviation, indicating greater variability. This suggests that while DQN tuning can enhance exploration strategies, it may also introduce inconsistency compared to the randomly tuned approach. In dynamic environments, both randomly tuned TS and DQN-TS outperform standard TS.

The effectiveness of randomly tuned Thompson Sampling in dynamic settings is due to its inherent randomness, which fosters exploration. The DQN's action space broadens reward distributions, encouraging greater exploration even after initial convergence. Consequently, this responsiveness to environmental changes allows it to outperform regular TS, though DQN-TS shows only a slight advantage over randomly tuned TS in cumulative regret. In summary, DQN tuning is more significant in UCB-based algorithms than in TS algorithms.

5 Conclusion

This paper explored the use of DQN to dynamically tune the exploration parameter in MAB algorithms, focusing on the UCB and TS frameworks in static and dynamic environments. Through extensive experiments on the MovieLens dataset, we showed that DQN-tuned UCB consistently outperformed standard UCB and randomly tuned variants in static and dynamic environments. The DQN's adaptive nature allowed it to start with high exploration values and gradually decline, demonstrating intelligent parameter tuning. In dynamic environments where reward distributions changed periodically, the DQN-based approach maintained strong performance by adjusting parameters in response to these shifts. The ablation study confirmed that the performance improvements were due to learning meaningful policies, not random parameter switching.

In contrast, DQN-tuned TS yielded mixed results. While it performed similarly to standard TS in static environments, with higher mean regret and variability, it showed significant improvement in dynamic settings, mirroring DQN-UCB by increasing exploration when changes were detected. However, the ablation study indicated that it did not outperform randomly tuned TS, which benefited from stochastic variability that aligned well with the dynamic environment.

Several directions could further improve the proposed framework. First, developing more informative and fine-grained state representations, especially for TS-based models. This could help the DQN distinguish between subtle environment shifts and achieve a better result. Second, experimenting with deeper or alternative neural network architectures, such as attention mechanisms or recurrent models, may enable more efficient learning in dynamic environments. Third, the applicability of this framework would be broadened by extending it to optimise parameters in other UCB or TS-based methods. Ultimately, this line of research aims to utilise AI models to establish a general, learning-based controller for exploration tuning across a wide range of bandit algorithms and real-world settings.

References

1. Slivkins, A.: ‘Introduction to Multi-Armed Bandits’Foundations and Trends® in Machine Learning, 2019, 12, (1-2), pp. 1–286.
2. Liu, Y., Roy, V., Xu, K.: ‘A Definition of Non-Stationary Bandits’arXiv (Cornell University), 2023.
3. Trovo, F., Paladino, S., Restelli, M., Gatti, N.: ‘Sliding-Window Thompson Sampling for Non-Stationary Settings’Journal of Artificial Intelligence Research, 2020, 68, pp. 311–364.
4. Liu, H.: ‘Comparative analysis of Sliding Window UCB and Discount Factor UCB in non-stationary environments: A Multi-Armed Bandit approach’Applied and Computational Engineering, 2024, 49, (1), pp. 136–141.
5. Ghiye, A., Barreau, B., Carlier, L., Vazirgiannis, M.: ‘Adaptive Collaborative Filtering with Personalized Time Decay Functions for Financial Product Recommendation’Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 798–804.
6. Gigli, M., Stella, F.: ‘Multi-armed bandits for performance marketing’International Journal of Data Science and Analytics, 2024.
7. Xiang, D., West, R., Wang, J., Cui, X., Huang, J.: ‘Multi Armed Bandit vs. A/B Tests in E-commerce - Confidence Interval and Hypothesis Test Power Perspectives’Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 4204–4214.
8. Cunha, T., Marchini, A.: ‘A Hybrid Meta-Learning and Multi-Armed Bandit Approach for Context-Specific Multi-Objective Recommendation Optimization’arXiv (Cornell University), 2024.
9. Deliu, N.: ‘Reinforcement learning for sequential decision making in population research’Quality & Quantity, 2023, 58, (6), pp. 5057–5080.
10. Bouneffouf, D., Claeys, E.: ‘Hyper-parameter Tuning for the Contextual Bandit’arXiv (Cornell University), 2020.
11. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: ‘Human-level Control through Deep Reinforcement Learning’Nature, 2015, 518, (7540), pp. 529–533
12. Harper, F.M., Konstan, J.A.: ‘The MovieLens Datasets’ACM Transactions on Interactive Intelligent Systems, 2015, 5, (4), pp. 1–19.