Deep Exploration for Recommendation Systems

Zheqing Zhu
billzhu@fb.com
Meta AI, Stanford University
Menlo Park, CA, USA

Benjamin Van Roy
bvr@stanford.edu
Stanford University
Stanford, CA, USA

ABSTRACT

Modern recommendation systems ought to benefit by probing for and learning from delayed feedback. Research has tended to focus on learning from a user’s response to a single recommendation. Such work, which leverages methods of supervised and bandit learning, forgoes learning from the user’s subsequent behavior. Where past work has aimed to learn from subsequent behavior, there has been a lack of effective methods for probing to elicit informative delayed feedback. Effective exploration through probing for delayed feedback becomes particularly challenging when rewards are sparse. To address this, we develop deep exploration methods for recommendation systems. In particular, we formulate recommendation as a sequential decision problem and demonstrate benefits of deep exploration over single-step exploration. Our experiments are carried out with high-fidelity industrial-grade simulators and establish large improvements over existing algorithms.

CCS CONCEPTS

• Information systems → Recommender systems; • Theory of computation → Sequential decision making.

KEYWORDS

Reinforcement Learning, Recommendation Systems, Decision Making under Uncertainty

ACM Reference Format:
Zheqing Zhu and Benjamin Van Roy. 2023. Deep Exploration for Recommendation Systems. In Seventeenth ACM Conference on Recommender Systems (RecSys ’23), September 18–22, 2023, Singapore, Singapore. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3604915.3608855

1 INTRODUCTION

Recommendation systems (RS) play a critical role in helping users find desired content within the ever-expanding universe of information that can be accessed via the Internet. Supervised learning procedures, such as collaborative filtering[35] and content-based filtering[3] that form the cornerstone of RS, are typically used to model the probability that each piece of content will immediately engage a given user. However, those methods are inadequate when optimizing for delayed user feedback [20, 22]. In particular, the value of a recommendation can become evident in later interactions with a user, rather than through immediate engagement. For example, a user might submit a positive rating about a news platform after a series of recommendations.

Delays in relevant feedback are prevalent in practical RS. One approach to developing a RS that accounts for delayed feedback is to maintain a model that predicts cumulative future value from interactions with each user, using reinforcement learning (RL) algorithms [6, 10, 18, 19, 24, 37, 40, 47]. Maximizing cumulative future value is especially important when making recommendations that can trigger informative feedback only after subsequent interactions with the user. Although the aforementioned work has demonstrated potential value of RL in RS, sparse and delayed feedback poses challenges to fully realizing the value of RL.

A major challenge to existing RS algorithms arises when positive feedback is rare and tends to arise only after a series of suitable recommendations. For the aforementioned methods to learn from such feedback, enormous quantities of data are required. Indeed, data requirements grow proportionately with the ratio between uninformative and informative feedback [11]. Recommendations that aim to elicit informative feedback can dramatically reduce data requirements. Some recent research focuses on bandit-based exploration algorithms such as upper-confidence bound (UCB) [27] and Thompson sampling [5] algorithms. However, such exploration strategies are myopic in the sense that they seek information only from immediate user feedback. Deep exploration [30] algorithms, on the other hand, further seek information revealed by sequences of rather than individual recommendations. This enables RS to more quickly learn users’ preferences even when their feedback is sparse and delayed. To the best of our knowledge, this paper is the first to demonstrate the importance of deep exploration in RS. We develop the first deep exploration algorithm for RS, leveraging randomized value functions [30] and epistemic neural networks (ENNs) [31].

2 RELATED WORK

Contextual Bandit (CB). One key advantage of adopting CB is to leverage the mechanism of exploration, where an agent seeks for useful information. The most common exploration strategy in RS is $\epsilon$-greedy [17, 21, 40]. $\epsilon$-greedy keeps a point estimate for the value of each recommendation and provides a random recommendation with probability $\epsilon$, which may or may not help learn more information about this recommendation. Beyond $\epsilon$-greedy, an agent can adopt the upper confidence bound (UCB) [1, 12] strategy, where the agent deems the upper bound of the reward of an arm as its belief of the environment and updates the confidence bound of each arm as it interacts with the environment. The strategy is extended to LinUCB [23] for logistic regression and neural networks [48]. Another branch of exploration strategy follows Thompson sampling [41], where an agent samples from its posterior belief of bandit arms’ rewards and selects the arm with highest sample estimate.
The algorithm is also extended to logistic regression [5] and neural network versions through ensemble sampling [25, 33] or neural Thompson sampling [32, 46]. However, both UCB and Thompson sampling for CB remain to explore myopically, i.e. only explore for immediate reward.

Reinforcement Learning (RL). Recent research in RS has expanded its focus beyond contextual bandits and starts to study the possible design of RL algorithms in RS. RL optimizes beyond immediate reward and maximizes cumulative reward. Using RL, we have seen significant user satisfaction improvements with massive datasets available for learning from offline RL with value function approximation [8, 14, 34, 39], policy gradient [6] and actor-critic [7], but none of the work above focuses on the data efficiency of their solutions and consumes a massive amount of data to produce a RL model. Exploration in deep RL goes beyond the exploration strategy in bandits. Agents leveraging RL aims to optimize long-term cumulative reward and hence needs to seek for information beyond immediate reward.

3 PROBLEM FORMULATION

In this section, we formulate RS as a sequential decision problem in which the agent interacts with a community of users. Each user is assigned a distinct positive integer index. At each time $t$, there is a set $U_t$ of active users. Any particular user may be in any combination of these sets across time. At each time, the agent registers an observation from and then provides a recommendation to each active user. We model interactions as generated by an environment, which is identified by a quintuple $E = (O, A, r, c, \rho)$.

1. **Observation set $O$**: At each time $t$, the agent receives an observation $O_{t,u}$ from each active user $u \in U_t$. The set $O$ is made up of all possible values of $O_{t,u}$.

2. **Action set $A$**: At each time $t$, the agent applies an action $A_{t,u}$ to each active user $u \in U_t$. The set $A$ is made up of all possible values of $A_{t,u}$.

3. **Reward function $r$**: The reward function generates a real-valued reward $R_{t,u} = r(O_{t,u})$ based on the observation $O_{t,u}$.

4. **Constraint function $c$**: It is common in RS that not all items are available to all users at all times. We model this in terms of a constraint function $c$, which prescribes a set of allowed recommendations. In particular, given an observation $O_{t,u}$, the corresponding action is constrained so that $A_{t,u} \in \mathcal{A}(O_{t,u}) \subseteq \mathcal{A}$.

5. **Observation probability function $\rho$**: Let $O_t = \{(u, O_{t,u}) \mid u \in U_t\}$ encode all the agent observes at time $t$, and let $A_t = \{(u, A_{t,u}) \mid u \in U_t\}$ encode all actions executed by the agent. Let $H_t = (O_0, A_0, O_1, \ldots, O_t) \in \mathcal{H}$ denote the history of the agent’s experience through time $t$. The observation probability function $\rho(\cdot \mid H_t, A_t) = \mathbb{P}(O_{t+1} = \cdot \mid H_t, A_t)$.

Agents will develop deep environments of the form we have described. The behavior of an agent can be characterized by a function $\pi_{agent}$ which, for any history $h$, specifies a probability mass function $\pi_{agent}(\cdot \mid h)$ over actions. The agent selects each action according to $A_t \sim \pi_{agent}(\cdot \mid H_t)$. To assess the performance of an agent in an environment $E = (O, A, r, c, \rho)$ over $T$ timesteps, we compute its cumulative reward $\sum_{t=0}^T \sum_{u \in U_{t+1}} R_{t+1,u}$. Note that this formulation is very general; interactions arising in most RS can be modeled in these terms.

4 WHY DEEP EXPLORATION?

Exploration is vital to sample efficiency when faced with sparse and delayed user feedback. Past research on RS has addressed exploration through either $\epsilon$-greedy schemes [17, 21, 40] in a RL context, or upper confidence bound (UCB) [26, 27] and Thompson sampling [4, 5] in a bandit learning context. The first approach randomly perturbs the exploitation strategy without strategically seeking useful information. The latter two approaches are guided by epistemic uncertainty of immediate feedback and lead to myopic exploration. However, actions of an intelligent agent ought to also depend on epistemic uncertainty about delayed feedback. We consider an approach that addresses this via estimating epistemic uncertainty about the optimal value function. Here, the optimal value function of a history-action pair $(h, a)$, denoted by $Q : \mathcal{H} \times \mathcal{A} \rightarrow \mathbb{R}$, refers to the expected reward an agent can accumulate by taking action $a$ at the history $h$ and adhering to an optimal policy thereafter.

Deep exploration refers to a strategic approach that positions an agent to more effectively gather information over subsequent time steps [30]. In this section, we investigate a deep exploration agent that maintains an approximation to the posterior distribution of the optimal value functions, samples a value function from this distribution upon engaging with a user and adheres to that sample until the user disengages. We refer to the period of user engagement as the user life-cycle. To evaluate the effectiveness of such an agent, we introduce SeqRec, which provides an example environment featuring sparse and delayed rewards. SeqRec represents a RS scenario in which users do not provide any feedback to the agent until the agent’s sequence of recommendations successfully meets their needs.

Example 1. SeqRec Environment

SeqRec features a fixed set of users, such that $U_t = U = \{1, 2, \ldots, N\}$ for all $t$. Specifically, SeqRec is defined by the following components, instantiating the environment quintuple $(O, A, r, c, \rho)$.

1. **The observation set $O = \{0, 1\}^2$. $O_{t,u} = (\text{Satisfied}_{t,u}, \text{Leave}_{t,u}) \in \mathcal{O}$, where $\text{Satisfied}_{t,u} \in \{0, 1\}$ indicates whether user $u$ is satisfied with the recommendation, and $\text{Leave}_{t,u} \in \{0, 1\}$ indicates whether the user intends to disengage with the RS agent. $O_0 = \{(u, (0, 0)) \mid u \in U_0\}$.**

2. **The action set $A = \{a_1, a_2, \text{no-op}\}$, where $a_1$ and $a_2$ are the only two genres of content for recommendation in the environment.**

3. **Reward $R_{t,u} = r(O_{t,u}) = \text{Satisfied}_{t,u}$.**

4. **The available set of actions $A_{t,u}$ at time step $t$ for user $u$ and the constraint function $c$ are defined as $A_{t,u} = \{a_1, a_2\}$, if $\text{Leave}_{t,u} = 0$, and $\{\text{no-op}\}$, if $\text{Leave}_{t,u} = 1$.**

5. **The observation probability mass function $\rho$ is described by the following mechanism. The environment tracks user satisfaction level $Y_{t,u} \in \mathbb{R}$ for every user $u \in U_t$. The environment uses an internal function, $g : \mathcal{U} \times \mathcal{A} \rightarrow \mathbb{R}$, to calculate the change in user satisfaction given an action. $Y_t+1,u = Y_t,u + g(u, A_{t,u})$, if $\text{Leave}_{t,u} = 0$, and 0 if $\text{Leave}_{t,u} = 1$.**
The environment, in addition, tracks each user $a$’s life-cycle length at time $t$ as $L_{t,u}$. $L_{t,u} = L_{t-1,u} + 1$, if $\text{Leave}_{t,u} = 1$, 0, if $\text{Leave}_{t,u} = 0$. Each user $u$ has a target satisfaction level $b_u$, where Satisfied$_{1,u} = 1$ if $Y_{t+1,u} > b_u$. A user decides to disengage with the agent, $\text{Leave}_{t+1,u} = 1$, when the user is satisfied with the agent, $Y_{t+1,u} \geq b_u$, the user’s satisfaction level drops below 0, $Y_{t+1,u} < 0$, or the user’s current life-cycle with the agent is longer than their engagement time budget $L_{t+1,u} = \tau_u, \tau_u \in \mathbb{N}$. Lastly we have the probability mass function $\rho$ as $\rho(O_{t+1} = \{(u, \text{Satisfied}_{t+1,u}, \text{Leave}_{t+1,u}) \mid u \in U_{t+1}\} \mid H_t, A_t) = 1$.

SeqRec represents an abstract and simplified model of user interaction. Delayed user feedback in this model is driven by the latent random variable $Y_{t,u}$ that tracks a users level of satisfaction. The environment, in addition, tracks each user $u$’s life-cycle to learn the optimal action. Delayed feedback is commonly observed in practical RSs arising in ratings on software applications, cumulative revenue, and total engagement time. In the following analyses, we will compare deep exploration against $\epsilon$-greedy, Thompson sampling with Gaussian priors, and UCB in a set of case studies using SeqRec.

### 4.1 SeqRec Sample Complexity: Single-User Case

Consider a SeqRec environment with a single user, $U = \{1\}$, with a satisfaction change function $g(1, a_1) = 0$ and $g(1, a_2) = 1$. The user’s satisfaction target level and engagement time budget are $b_1 = \tau_1 = T$. The agent’s goal is to maximize cumulative reward in SeqRec. The agent has prior knowledge that only one of $a_1$ and $a_2$ increases the user’s satisfaction level.

The optimal approach here is to consistently apply either action $a_1$ or $a_2$ until the user disengages, in order to determine which action leads to an increased satisfaction level. If the optimal strategy is not employed, the agent will not receive any positive rewards before the user disengages. In the following analysis, we demonstrate the difference between $\epsilon$-greedy, myopic Thompson sampling, myopic UCB and deep exploration.

Since the agent cannot observe any differences until it applies action $a_2$ for $T$ times, there is a unique sequence of recommendations that will reveal the optimal action. Employing $\epsilon$-greedy, the expected number of life-cycles required to learn the optimal policy is $\Theta(2^T)$, as there are $2^T$ possible permutations of recommendation sequences in $T$ steps.

Consider a myopic Thompson sampling bandit learning agent with zero-mean Gaussian priors. Given a history $h$, the agent samples from the posterior distributions over the immediate reward and selects the action with the higher reward sample. Since both actions’ immediate reward posteriors are zero-mean Gaussians before receiving any positive feedback, there is a $1/2$ probability of choosing either action $a_1$ or $a_2$. This probability remains constant after each posterior update until the desired sequence, applying action $a_2$ for $T$ times, is discovered. Thus, the expected number of user life-cycles with Thompson sampling is also $\Theta(2^T)$.

For the myopic UCB strategy, a common approach uses $\sqrt{\log(t)/N_t(a)}$ as the exploration bonus, in addition to the marginal reward estimate for a history-action pair, where $N_t(a)$ denotes the number of times action $a$ is executed up to timestep $t$. When the user starts to engage, the agent is equally optimistic about $a_1$ and $a_2$, with identical reward estimates. After each step, the agent reduces its exploration bonus for the selected action, ensuring that the same action will not be chosen in the next timestep. Consequently, the agent continuously alternates between the two actions and never achieves the desired sequence using UCB.

With our approach to deep exploration, the agent maintains an approximate posterior distribution of the optimal value functions $Q$. Initially, the agent’s prior belief assigns a $1/2$ probability to either $Q(h, a_1) = 1$ or $Q(h, a_2) = 1$ for any $h$ that consistently applies $a_1$ or $a_2$, respectively, and assigns a value of 0 to all other history-action pairs. During any user life-cycle prior to receiving positive feedback, there is a $1/2$ probability that the agent will consistently apply action $a_2$ for $T$ times, thereby eliminating the possibility of $Q(h, a_1) = 1$. In expectation, the agent needs 2 lifecycles to determine $Q(h, a_2) = 1$ when persistently applying $a_2$. Consequently, the agent requires only $\Theta(1)$ user life-cycle to learn the optimal policy using the deep exploration strategy.

### 4.2 SeqRec Sample Complexity: Multi-User Case

Consider a SeqRec environment with $N$ users, denoted as $U = \{1, 2, \ldots, N\}$. For some users in $U$, action $a_1$ increases satisfaction by 1, while for others, action $a_2$ does the same. For each user $u \in U$, the action that improves satisfaction is deterministic, and we refer to this action as user $u$’s preferred action. We have $b_u = \tau_u = T$ for all $u \in U$, and $N \ll 2^T$. The agent aims to optimize cumulative rewards across all users, with prior knowledge that either action $a_1$ or $a_2$ enhances each user’s satisfaction level. We assume a function approximator capable of accurately modeling each user’s preferred action after observing the preferred actions of $N/2$ users.

We extend Analysis 1 to a multi-user setting, where learnings from one user can be generalized to serve other users as well. Our aim is to demonstrate that deep exploration remains crucial to an agent’s success in a RS environment with a large user base, even when equipped with powerful generalization capabilities. Since the myopic UCB strategy is unable to identify the optimal policy even in single-user SeqRec, we exclude it from this analysis.

First, we consider the $\epsilon$-greedy and myopic Thompson sampling strategies. As an agent leveraging such strategies have a $\Theta(1/2^T)$ probability of learning the preferred action for any user $a$ in a life-cycle, the expected number of preferred actions learned after all $N$ users’ first life-cycles is $\Theta(N/2^T)$. In expectation, $\Theta(2^{T-1})$ life-cycles from all $N$ users are needed for the agent to learn all preferred actions. If the agent undergoes $M$ life-cycles and $M \leq 2^{T-1}$, the expected cumulative reward across the $N$ users in $U$ is $\Theta(MN/2^T)$ and $\Theta(N(M - 2^{T-1}))$ if $M > 2^{T-1}$.

Now, let’s consider deep exploration. This algorithm has a $1/2$ probability of learning each user’s preferred action after a single life-cycle of interaction. In expectation, after the first life-cycles of users in $U$, the system can perfectly model all users’ preferred actions, yielding an expected cumulative reward of $\Theta(NM)$ in $M$ life-cycles.

Analyses above highlight that intelligent exploration is crucial for learning both single-user and population preferences. $\epsilon$-greedy, myopic Thompson sampling, and myopic UCB, are ill-suited for
problems characterized by sparse and delayed consequences, even when paired with function approximators possessing strong generalization capabilities like deep neural networks. In contrast, deep exploration is much more adept at handling these types of problems.

5 EPISTEMIC UNCERTAINTY ESTIMATION VIA EPISTEMIC NEURAL NETWORKS (ENN)

Advanced models, such as Neural Collaborative Filtering [16], Wide & Deep [9], and Multi-Head Self-Attention [43], build sophisticated representations for users, recommendations, and their interactions. However, they are unable to identify what they do not know, i.e., assessing the epistemic uncertainty concerning the users’ preferences. In this section, we will focus on two neural networks (NN) for estimating epistemic uncertainty to carry out deep exploration.

1) Deep Ensemble [25, 28, 29]. Deep ensemble employs an ensemble of NNs to estimate the posterior distribution of a target variable. This architecture is initialized with $M$ base NNs, denoted as $\{f_{\beta_1}, \ldots, f_{\beta_M}\}$, with each network parameterized by $\beta_1, \ldots, \beta_M$, respectively. Additionally, $M$ prior NNs, $\{f_{\theta^p_1}, \ldots, f_{\theta^p_M}\}$, are initialized using Glorot sampling and remain unaltered throughout the model’s lifespan [13, 28]. These prior NNs serve to regularize the posterior parameters, preventing the posterior distribution from collapsing too easily [2, 45]. For a given input $x \in \mathbb{R}^d$, a deep ensemble generates a posterior sample estimate by sampling $z \sim \text{Unif}(1, \ldots, M)$ and computing $\hat{y} = f_{\beta_z}(x) + \alpha f_{\theta^p_z}(x)$. Here, $f : \mathbb{R}^d \rightarrow \mathbb{R}$, and $\alpha = (0, 1)$ is a scaling factor. This method is known as ensemble sampling [25], which approximates Thompson sampling [33]. Given a loss function $\mathcal{L}$ and a target variable $y$, an ensemble optimizes each particle $\beta_z$,

$$\beta_z = \arg \min_{\beta} \mathcal{L}(y, f_{\beta}(x) + \alpha f_{\theta^p}(x)).$$

For simplicity in notation, we denote $h_{\theta, \theta^p}(x, z) = f_{\beta_z}(x) + \alpha f_{\theta^p_z}(x)$ as the $z$th neural network in the ensemble, where $h : \mathbb{R}^d \times \{1, \ldots, M\} \rightarrow \mathbb{R}$ and $\theta = (\beta_1, \ldots, \beta_M), \theta^p = (\theta^p_1, \ldots, \theta^p_M).

2) EpiNet [31]. Although deep ensemble provides a valuable tool for epistemic uncertainty estimation, its scalability with large deep NNs remains a challenge. There are two primary bottlenecks of deep ensembles. First, the number of parameters required is $M$ times larger than that of a single NN. Second, a deep ensemble can only offer $M$ particles sampled from the approximate posterior distribution. EpiNet [31] proposes a potential solution to both bottlenecks.

EpiNet is an add-on architecture designed for general NN architectures, enabling epistemic uncertainty estimation and improved joint predictions. Given a NN function approximator $f_{\beta} : \mathbb{R}^d \rightarrow \mathbb{R}$, parameterized by $\beta$, EpiNet first extracts the last layer representation $\sigma_{\beta}(x)$, where $\sigma_{\beta} : \mathbb{R}^d \rightarrow \mathbb{R}^d$. Next, EpiNet concatenates the representation with an epistemic index $z \in \mathbb{R}^d$ and executes a forward pass on the concatenated vector. The epistemic index can be a one-hot vector or sampled from a Gaussian prior. EpiNet also maintains a main NN $g_{\theta}$ and a prior NN $g_{\theta^p}$, where $g : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$. Hence given a NN $f_{\beta}$, an input feature vector $x$ and an epistemic index $z$, the posterior sample is

$$\hat{y} = f_{\beta}(x) + \left\{ g_{\theta}(\sigma_{\beta}(x)), z \right\} + \alpha g_{\theta^p}(\sigma_{\theta^p}(x)), z \right\}^T z. \quad (1)$$

where $\text{sgl}[-]$ indicates stop-gradient in backpropagation. In a logistic regression case, $\hat{y}$ goes through an additional Sigmoid function.

For simplicity of notation and consistency with the notations for deep ensemble, we denote Equation 1 by $h_{\theta, \theta^p}(x, z)$, where $\theta = (\beta, \eta), \theta^p = \eta^p$. During optimization, given a target variable $y$,

$$\theta = \arg \min_{\theta} \sum_{z \in \mathcal{Z}} \mathcal{L}(y, h_{\theta, \theta^p}(x, z)).$$

Here $\mathcal{Z}$ is a set of epistemic indices sampled from $z$’s prior to approximate the loss across the prior distribution of $z$.

6 A DEEP EXPLORATION AGENT

In this section, we present an efficient implementation of deep exploration utilizing Randomized Value Function (RVF) [30] through ENNs and Deep Q-learning. RVF exhibits two main features. First, it perturbs rewards, encouraging exploratory behavior. Second, an RVF agent consistently adheres to the policy induced by the value function sampled from an ENN for the entire user lifecycle, facilitating deep exploration. We denote the representation of a user as $\psi_u \in \mathbb{R}^d$ and that of an action as $a \in \mathcal{A}$. Both $\psi_u$ and $\phi_a$ are deterministic and remain constant over time. Additionally, the agent is offered extract_interact_features : $\mathcal{H} \times \mathcal{U} \rightarrow \mathbb{R}^d$, where $\xi_{t,u} = \text{extract_interact_features}(h_t, u)$ represents the historical interaction between user $u$ and the agent up to timestep $t$.

6.1 Deep Q-learning

In the context of RS, a deep Q-learning agent uses a NN to estimate its value function, $Q_{\theta} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, parameterized by $\theta$. Following the problem definition in Section 3, at time step $t$, a deep Q-learning agent selects an action:

$$A_t = \left\{ u, \arg \max_{a \in \mathcal{A}_{t,u}} Q_{\theta}(\psi_{u}, \phi_a, \xi_{t,u}) : u \in \mathcal{U}_t \right\},$$

where $\mathcal{A}_{t,u} = c(O_{t,u})$.

The agent explores by either modifying $Q_{\theta}$ or adopting a random exploration approach. For instance, when using $c$-Greedy as the exploration strategy, with probability $c$, the agent chooses a random action and follows the value function greedily otherwise. After executing action $A_t$, the agent receives a new observation $O_{t+1}$ from the environment and computes reward $R_{t+1}$. If $u \neq \mathcal{U}_t \cap \mathcal{U}_{t+1}$, the agent stores a transition $(\psi_{t,u}, \phi_a, \xi_{t,u}, R_{t+1}, \xi_{t+1,u}, A_{t+1,u})$, where $\mathcal{A}_{t+1,u} = c(O_{t+1,u})$, in its replay buffer $\mathcal{D}$ for future learning. It is worth noting that this approach can be easily extended to accommodate multi-step transitions by referencing the last observation generated by user $u$.

After several interactions, the agent samples data from its replay buffer to improve its value function model $Q_{\theta}$. To stabilize learning, a deep Q-learning agent creates a copy of the value function model as the target model $Q_{\theta^p}$ and copying the parameters from $Q_{\theta}$ to $Q_{\theta^p}$ every $K$ steps. More formally, with learning rate $\alpha$, the update
for $\beta$ is:

$$\beta \leftarrow \beta - \alpha \nabla \beta \sum_{u \in \mathcal{U} \cap \mathcal{U}_{t+1}} \left[ (R_{t+1,u} + \max_{a \in \mathcal{A}_{t+1,u}} Q^\beta (\psi_u, \phi_{A_{t+1,u}, \xi_{t+1,u}}) - Q^\beta (\psi_u, \phi_{A_t,u}, \xi_{t,u}))^2 \right] |_{\beta=\beta_n} .$$

(2)

We omitted the discount factor, common in RL literature, as we are optimizing for fixed-horizon cumulative reward.

### 6.2 Deep Exploration via Randomized Value Function (RVF) and Epistemic Neural Network (ENN)

An agent can conduct deep exploration by employing Deep Q-learning through RVF and an ENN, $h_Q^{\theta, \theta_p}$, in order to estimate the posterior distribution of value functions. $h_Q^{\theta, \theta_p}$ takes a user representation, an action representation, an interaction feature vector and an epistemic index as input and outputs a value function sample from an approximate posterior distribution. At the beginning of a user life-cycle, the agent samples an epistemic index $z_0 \in \mathbb{R}^d_z$ from the prior distribution $P_z$, and maintains this index throughout the life-cycle. At timestep $t$, the agent greedily selects:

$$A_t = \left\{ u, \arg \max_{a \in \mathcal{A}_{t,u}} h_Q^{\theta, \theta_p} (\psi_u, \phi_a, \xi_{t,u}, z_u) : u \in \mathcal{U}_t \right\} .$$

(3)

See Section 5 for two potential architectures for $h_Q^{\theta, \theta_p}$. After taking $A_t$, the agent observes $O_{t+1}$ and computes $R_{t+1}$.

A deep exploration agent also maintains replay buffers. To incentivize exploration, we adopt the reward perturbation approach in [30] when updating the buffers. There is a notable distinction when using a deep ensemble compared to an EpiNet: an agent utilizing a deep ensemble must maintain a separate perturbed buffer for each particle NN within the ensemble, whereas an agent employing an EpiNet requires only a single perturbed buffer. To elaborate, if $u \in \mathcal{U}_t \cap \mathcal{U}_{t+1}$, an agent leveraging deep ensemble stores a transition $(\psi_u, \phi_{A_{t+1,u}}, \xi_{t,u}, R_{t+1,u} + W_{z}^{x_{t+1,u}, \xi_{t+1,u}, \mathcal{A}_{t+1,u}})$ in the zth buffer for future learning. Here, $\mathcal{A}_{t+1,u} = c(O_{t+1,u}), W_{z}^{x_{t+1,u}} \sim N(0, \sigma^2)$ and $\sigma^2$ is a hyperparameter. On the other hand, an agent employing an EpiNet only maintains a single replay buffer and stores a single transition with $W_{t+1,u} \sim N(0, \sigma^2)$.

The agent then samples from its buffer to update its parameter $\theta$. With learning rate $\alpha$, $\theta$ is updated via

$$\theta \leftarrow \theta - \alpha \nabla \theta \sum_{z \in Z} \sum_{u \in \mathcal{U}_t \cap \mathcal{U}_{t+1}} \left[ \tilde{R}_{t+1,u,z} + \right.$$  

$$\left. \max_{a \in \mathcal{A}_{t+1,u}} h_Q^{\theta, \theta_p} (\psi_u, \phi_a, \xi_{t+1,u,z}) - h_Q^{\theta, \theta_p} (\psi_u, \phi_{A_{t+1,u}, \xi_{t,u,z}})^2 \right] |_{\theta=\theta_n} .$$

(4)

Here, $\tilde{R}_{t+1,u,z}$ denotes the perturbed reward and $h_Q^{\theta, \theta_p}$ represents the target network. For an agent utilizing EpiNet, $Z$ is a set of epistemic indices sampled from $z$’s prior, which helps approximate the loss across the entire distribution of $z$. Moreover, $\tilde{R}_{t+1,u,z}$ remains consistent across all $z$ values. Conversely, for an agent that employs a deep ensemble, $\tilde{R}_{t+1,u,z}$ corresponds to the perturbed reward from the zth replay buffer in the ensemble, and $Z$ is defined as $\{1, \ldots, M\}$. For a comprehensive view of the deep exploration algorithm with EpiNet, please refer to Figure 1 for visualization.

### 7 EXPERIMENTS

In this section, we conduct a series of experiments to demonstrate the benefits of deep exploration. We present a toy experiment, an experiment based on an e-commerce environment, and another experiment for slate ranking with parallel user streams. These environments, designed by industry leaders, aim to ensure high-fidelity, model-based evaluation grounded on real-world data. We compare deep exploration agents based on both deep ensemble and EpiNet against Q-learning agents that utilize strategies such as ε-greedy, Neural Thompson Sampling (Neural TS) [46], Neural UCB [48], and Neural LinUCB [44]. In all experiments, the scaling factor of the prior network is set to 0.3. Deep ensemble’s cardinality is set to 10. For EpiNet, the number of epistemic indices during optimization is set to 50, with their dimension set to 10, drawn from a standard Gaussian prior. All agents share the same NN architecture, featuring a single hidden layer of 20 units for the toy experiment, and hidden layers (200, 100, 50, 25) for the remaining experiments. All experiments are executed on a cloud machine with 2 A100 GPUs. The results are presented in Table 1.

1) **SeqRec Experiment.** Following SeqRec in Example 1, we set $\mathcal{U}_t = 1, v_t$, with $v_1 = 10, b_1 = 1, g(1, a_1) = 0$, and $g(1, a_2) = 1$. Contrary to Analysis 1, we do not implicitly inject prior knowledge into the agent. The action features are provided to the agent as one-hot feature vectors and each entry of the interaction feature $\xi_{t,u}$ is defined as: $\xi_{t,u}[i] = -1$, if $i > L_{t,u}, 1$ if $A_{t-L_{t+u+1,u}} = a_1$ at time step $t = L_t + i$, and 0 otherwise, where $L_{t,u}$ is the user’s current life-cycle length. The results of the experiment, averaged over ten random seeds each with 100 life-cycles, are presented in the first column of Table 1. The results indicate that only deep exploration is able to learn a reasonable policy.

2) **E-Commerce Environment Experiment.** Virtual-Taobao [36], a model-based simulator on one of the world’s largest E-Commerce platforms, allows us to evaluate agent performance with high fidelity. The environment delivers user feedback and termination via Multi-Agent Adversarial Imitation Learning (MAIL) [38]. At the onset of each episode, the simulator samples a user from the user distribution via a Generative Adversarial Network (GAN) [15]. At each time step, the agent is presented with 100 recommendation candidates. The agent’s task is to select one to
Table 1: Average Life-Cycle Cumulative Reward across Users (DE Stands for Deep Exploration)

| Algorithm          | Toy E-Commerce Train | E-Commerce Eval | Multi-User Slate Ranking Train | Multi-User Slate Ranking Eval |
|--------------------|----------------------|-----------------|-------------------------------|------------------------------|
| \(\epsilon\)-Greedy | 0.0 ± 0.0            | 1.07 ± 0.02     | 0.95 ± 0.07                   | 94.79 ± 1.21                 |
| Neural TS          | 0.0 ± 0.0            | 6.55 ± 1.14     | 6.37 ± 0.90                   | 111.323 ± 1.34               |
| Neural UCB         | 0.0 ± 0.0            | 9.13 ± 0.04     | 8.035 ± 0.15                  | 117.341 ± 2.03               |
| Neural LinUCB      | 0.0 ± 0.0            | 9.42 ± 0.13     | 8.78 ± 0.17                   | 121.344 ± 4.40               |
| Ensemble - DE     | 0.59 ± 0.03          | 11.46 ± 0.83    | 9.82 ± 0.50                   | 135.844 ± 3.62               |
| EpiNet - DE       | 0.71 ± 0.15          | 16.64 ± 1.04    | 13.83 ± 0.25                  | 147.703 ± 7.11               |

(a) Average Life-Cycle Return for High-Fidelity E-Commerce Environment through Virtual-Taobao, 200 Users Each Serving 10 Times Sequentially

(b) Average Life-Cycle Return for High-Fidelity E-Commerce Environment with 200 Cold Start Users that Agent Has Never Seen Before

(c) Tradeoff between Life-Cycle Return and Computation for E-Commerce Experiment. Better Tradeoff towards Top Right.

(d) Average Life-Cycle Return for High-Fidelity Multi-User Slate Ranking Environment, 100 Users Each Served 10 Times with 10-User Parallel Streams

(e) Average Life-Cycle Return for High-Fidelity Multi-User Slate Ranking Environment with 100 Out of Sample Users in 10-User Parallel Streams

(f) Tradeoff between Life-Cycle Return and Computation for Multi-User Slate Ranking Experiment. Better Tradeoff towards Top Right.

Figure 2: Experiment Results across E-Commerce and Multi-User Slate Ranking Environments. Average life-cycle return in each of the figures above plots: \(\frac{1}{|U|} \sum_{u \in U} \sum_{t=1}^{T} R_{t,u}\) averaged over 30 distinct seeds, where \(U = \bigcup_{t=1}^{T} U_t\) is the community of all users the agent has served and \(R_{t,u}\) is the reward at time step \(t\) for user \(u\). Note that since the Eval set is a completely new set of users, it is normal to see evaluation’s average life-cycle return higher than training.

propose to the user. Additionally, the agent has access to feature vectors for users, recommendations and interaction history. Due to the computational limitations of neural Thompson sampling and neural UCB — both requiring inverting covariance matrices with the width of the NN’s full parameter size — we only utilize the last layer’s gradients for these methods.

Refer to Fig. 2a, which displays the results of an experiment involving 30 seeds serving 200 users, each engaged 10 life-cycles with the agent with each life-cycle happening sequentially. It is clear that both deep exploration strategies surpass other exploration methods. Notably, the deep exploration methods achieve reasonable performance much more rapidly than myopic exploration methods. We also evaluate the performance of agents trained online with 200 out-of-sample users, without any additional training. Please refer to Figure 2b. The deep exploration methods, particularly the EpiNet-based deep exploration, significantly outperforms all other candidates in out-of-sample evaluations. Finally, the computation-performance trade-off of the algorithms is illustrated in Fig. 2c,
where candidates situated towards the top right exhibit the optimal trade-off criteria. Here, again, the deep exploration strategies demonstrate a clear advantage.

3) Slate Ranking Experiment with Parallel User Streams.

We leverage RL4RS [42], a real-world simulator that provides parallel user streams and slate ranking. User behavior predictions are generated via the Deep Interest Evolution Network (DIEN) [49], and users are sampled from a real dataset with anonymized representations. The agent interacts with 10 users simultaneously, providing each user with a separate slate of recommendations. When a user concludes their interactions with the agent, the environment samples a new user to maintain a constant stream of 10 active users. At each time step, the agent is presented with 284 different combinations (slates) of recommendation items as its available actions. We apply the same treatment to NeuraTS and NeuraUCB as the E-Commerce Experiment. Refer to Fig. 2d for a 30-seed experiment serving 100 users, each with 10 life-cycles. The results illustrate that both deep exploration strategies outperform other exploration methods. On a set of 100 out-of-sample users, as shown in Fig. 2b, we observe that both deep exploration methods significantly outperform others. Lastly, we provide a visualization of the computation-performance tradeoff in Figure 2c and again both deep exploration strategies lead the competition.

8 CONCLUSION

In this paper, we addressed the issue of sparse and delayed feedback in RS and explored the potential for deep exploration to improve personalization. We formally defined the RS problem in the context of sequential decision making and presented real-world experimental results with deep exploration in different high-fidelity RS environments. We demonstrated promise, through these experiments, in improving data efficiency and amplifying the rate of positive feedback in a scalable manner relative to other exploration designs. We hope that this paper will inspire adoption of deep exploration and show impact on real-world platforms through data-efficient personalization.

REFERENCES

[1] Peter Auer, Nicolas Cesa-Bianchi, and Paul Fischer. 2002. Finite-Time Analysis of the Multiarmed Bandit Problem. Machine Learning, 47, 2-3 (2002), 235–256.
[2] Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. 2017. Spectrally-Normalized Margin Bounds for Neural Networks. In Advances in Neural Information Processing Systems. 6240–6249.
[3] Stephanie Blanda. 2016. Online Recommender Systems—How Does a Website Know What I Want? American Mathematical Society. Retrieved October 31 (2016).
[4] Bjorn Brodin, Mikael Hammar, Bengt J Nilsson, and Dimitris Paroschakas. 2018. Ensemble Recommendations via Thompson Sampling: an Experimental Study within E-Commerce. In 23rd international conference on intelligent user interfaces. 19–29.
[5] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In Advances in Neural Information Processing Systems. 2249–2257.
[6] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Bellelli, and Ed H Chi. 2019. Top-K Off-policy Correction for a REINFORCE Recommender System. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. 456–464.
[7] Minmin Chen, Can Xu, Vince Gatto, Devandum Jain, Aviral Kumar, and Ed Chi. 2022. Off-Policy Actor-critic for Recommender Systems. In Proceedings of the 16th ACM Conference on Recommender Systems. 338–349.
[8] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative Adversarial User Model for Reinforcement Learning based Recommendation System. In International Conference on Machine Learning. PMLR, 1052–1061.
[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Krish Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st workshop on deep learning for recommender systems. 7–10.
[10] Guadriel Desinena, Armando Diaz, Jalil Desinena, Ismael Morenon, and Daniel Garcia. 2019. Maximizing Customer Lifetime Value using Stacked Neural Networks: An Insurance Industry Application. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 541–544.
[11] Aarti Garganvar. 2012. An Overview of Classification Algorithms for Imbalanced Datasets. International Journal of Emerging Technology and Advanced Engineering, 2, 4 (2012), 42–47.
[12] Aurélien Garivier and Olivier Cappé. 2011. The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond. In Proceedings of the 24th annual conference on learning theory. JMLR Workshop and Conference Proceedings, 359–376.
[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. 249–256.
[14] Nick Golovin and Erhard Rahm. 2004. Reinforcement Learning Architecture for Web Recommendations. In International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004., Vol. 1. IEEE, 398–402.
[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. Commun. ACM 63, 11 (2020), 139–144.
[16] Xiangnan He, Xiaoyu Du, Xiang Wang, Peng Tian, Jinshi Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2227–2233.
[17] Binbin Hu, Chuan Shi, and Jian Liu. 2017. Playlist Recommendation Based on Reinforcement Learning. In International Conference on Intelligence Science. Springer, 172–182.
[18] Eugene Ie, Vihan Jain, Jing Wang, Sammut Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutillier. 2018. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets. In Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI-19). Macau, China, 2592–2599. see arXiv:1905.12767 for a related and expanded paper (with additional material and authors).
[19] Tomoharu Iwata, Kazumi Saito, and Takeshi Yamada. 2008. Recommendation Method for Improving Customer Lifetime Value. IEEE Transactions on Knowledge and Data Engineering, 20, 9 (2008), 1254–1263.
[20] Porous Joulaini, Andras Gyorgy, and Csaba Szepesvari. 2013. Online Learning under Delayed Feedback. In International Conference on Machine Learning. 1453–1461.
[21] Toshihiro Kamishima and Shotaro Akaho. 2011. Personalized Pricing Recommender System: Multi-stage Epsilon-Greedy Approach. In Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems. 57–64.
[22] Sofia Ira Itena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilip-kumar, Ferenc Hussar, Steven Yoo, and Wenzhe Shi. 2019. Addressing Delayed Feedback for Continuous Training with Neural Networks in CTR Prediction. In Proceedings of the 18th ACM Conference on Recommender Systems. 187–195.
[23] Li Hong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In Proceedings of the 19th International Conference on World Wide Web. 661–670.
[24] Duen-Ren Liu and Ya-Yueh Shih. 2005. Integrating AHP and Data Mining for Product Recommendation Based on Customer Lifetime Valsue. Information & Management, 43, 5 (2005), 387–400.
[25] Xiaoyuan Lu and Benjamin Van Roy. 2017. Ensemble Sampling. In Advances in Neural Information Processing Systems. 3258–3266.
[26] Atsuyoshi Nakamura. 2015. A UCB-like Strategy of Collaborative Filtering. In Proceedings of the 19th International Conference on World Wide Web. 661–670.
[27] Nhan Nguyen-Thanh, Dana Marinca, Kinda Khawam, David Rohde, Flavian Vasile, Elena Lohan, Steven Martin, and Dominique Quadri. 2019. Recommendation System-Based Upper Confidence Bound for Online Advertising. In REVEAL 2019.
[28] Ian Osband, John Aslanides, and Albin Cassirer. 2018. Randomized Prior Functions for Deep Reinforcement Learning. In Advances in Neural Information Processing Systems. 8617–8629.
[29] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep Exploration via Bootstrapped DQN. In Advances in Neural Information Processing Systems.
[30] Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. 2019. Deep Exploration via Randomized Value Functions. Journal of Machine Learning Research, 20, 124 (2019), 1–62.
[31] Ian Osband, Zheng Wen, Mohammad Ashgari, Morteza Ibrahimi, Xiyuan Lu, and Benjamin Van Roy. 2021. Epistemic Neural Networks. arXiv preprint arXiv:2107.08924 (2021).
[32] Ian Osband, Zheng Wen, Seyed Mohammad Ashgari, Vikranth Dwarkacherla, Xiuyuan Lu, Morteza Ibrahimi, Dieterich Lawson, Botao Hao, Brendan O’Donoghue, and Benjamin Van Roy. 2022. The Neural Testbed: Evaluating Joint Predictions in Advances in Neural Information Processing Systems.
[33] Chao Qin, Zheng Wen, Xiuyuan Lu, and Benjamin Van Roy. 2022. An Analysis of Ensemble Sampling. In Advances in Neural Information Processing Systems.

[34] Pornthep Rojanavasu, Phaithoon Sirimal, and Duen Pinnaple. 2005. New Recommendation System Using Reinforcement Learning. Special Issue of the Intl. J. Computer, the Internet and Management 13, SP 3 (2005).

[35] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative Filtering Recommender Systems. The Adaptive Web (2007), 291–324.

[36] Jing-Cheng Shi, Yang Yu, Qing Da, Shiyong Chen, and An-Xiang Zeng. 2019. Virtual-Taobao: Virtualizing Real-World Online Retail Environment for Reinforcement Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 4902–4909.

[37] Ya-Yueh Shih and Duen-Ren Liu. 2008. Product Recommendation Approaches: Collaborative Filtering via Customer Lifetime Value and Customer Demands. Expert Systems with Applications 35, 1-2 (2008), 350–360.

[38] Youming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. 2018. Multi-Agent Generative Adversarial Imitation Learning. In Advances in neural information processing systems.

[39] Xueying Tang, Yunxiao Chen, Xiaoui Liu, and Zhihong Ying. 2019. A Reinforcement Learning Approach to Personalized Learning Recommendation Systems. Brit. J. Math. Statist. Psych. 72, 1 (2019), 108–135.

[40] Georgios Theocharous, Philip S Thomas, and Mohammad Ghamvamzadeh. 2015. Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees. In Twenty-Fourth International Joint Conference on Artificial Intelligence.

[41] William R Thompson. 1933. On the Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. Biometrika 25, 3/4 (1933), 285–294.

[42] Kai Wang, Zhenhe Zou, Qin Deng, Yue Shang, Minghao Zhao, Runze Wu, Xudong Shen, Tangjie Lyu, and Changjie Fan. 2021. RL4RS: A Real-World Benchmark for Reinforcement Learning based Recommender System. arXiv preprint arXiv:2110.11073 (2021).

[43] Chuhua Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural News Recommendation with Multi-Head Self-Attention. In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). 6389–6394.

[44] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. 2021. Neural Contextual Bandits with Deep Representation and Shallow Exploration. In International Conference on Learning Representations.

[45] C Zhang, S Bengio, M Hardt, B Recht, and O Vinyals. 2017. Understanding Deep Learning Requires Rethinking Generalization Int. In Conf. on Learning Representations.

[46] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural Thompson Sampling. In International Conference on Learning Representations.

[47] Guanjie Zheng, Fuqiang Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 167–176.

[48] Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural Contextual Bandits with UCB-Based Exploration. In International Conference on Machine Learning. PMLR, 11492–11502.

[49] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaqing Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In Proceedings of the AAAI conference on artificial intelligence, Vol. 33. 5941–5948.