Exploring Multi-Objective Exercise Recommendations in Online Education Systems

Zhenya Huang[1], Qi Liu[1], Chengxiang Zhai[2,*], Yu Yin[1], Enhong Chen[1,*], Weibo Gao[1], Guoping Hu[3]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology & School of Data Science, University of Science and Technology of China,
{huangzhy,yxonic}@mail.ustc.edu.cn; {qiliuql,cheneh}@ustc.edu.cn; iamwebgao@gmail.com
[2]University of Illinois at Urbana-Champaign, czhai@illinois.edu; [3]iFLYTEK Research, gphu@iflytek.com

ABSTRACT

Recommending suitable exercises to students in an online education system is highly useful. Existing approaches usually rely on machine learning techniques to mine large amounts of student interaction log data accumulated in the systems to select the most suitable exercises for each student. Generally, they mainly aim to optimize a single objective, i.e., recommending non-mastered exercises to address the immediate weakness of students. While this is a reasonable objective, there exist more beneficial multiple objectives in the long-term learning process that need to be addressed including *Review & Explore*, *Smoothness of difficulty level* and *Engagement*. In this paper, we propose a novel *D*eep *R*einforcement learning framework, namely DRE, for adaptively recommending *E*xercises to students with optimization of above three objectives. In the framework, we propose two different Exercise Q-Networks for the agent, i.e., EQNM and EQNR, to generate recommendations following Markov property and Recurrent manner, respectively. We also propose novel reward functions to formally quantify those three objectives so that DRE could update and optimize its recommendation strategy by interactively receiving students' performance feedbacks (e.g., score). We conduct extensive experiments on two real-world datasets. Experimental results clearly show that the proposed DRE can effectively learn from the student interaction data to optimize multiple objectives in a single unified framework and adaptively recommend suitable exercises to students.

KEYWORDS

recommendation; deep reinforcement learning; multiple objectives

ACM Reference Format:
Zhenya Huang, Qi Liu, Chengxiang Zhai, Yu Yin, Enhong Chen, Weibo Gao, Guoping Hu. 2019. Exploring Multi-Objective Exercise Recommendations in Online Education Systems. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3-7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages.
https://doi.org/10.1145/3357384.3357995

*Corresponding Author.

1 INTRODUCTION

Online intelligent education, as an extension of traditional classroom setting, can leverage advanced technologies to provide personalized learning experiences to students [1]. Recent years have witnessed the proliferation of online education platforms, such as massive online open courses (MOOC), Khan Academy and online judging (OJ) systems [1, 8, 39]. They assist students with abundant learning materials (e.g., exercises), enabling them to learn and practice on their own pace according to the individual needs.

To achieve this goal, recommender systems play a crucial role, which helps students acquire knowledge by suggesting suitable exercises instead of letting them self-seeking [25]. They establish an open environment where students adaptively learn with the system agent individually. As illustrated in Figure 1, this procedure can be regarded as a set of interactions. At each time, the agent first presents an exercise to the student who then learns and answers it. Next, the agent receives performance feedback (e.g., right or wrong) from her and recommends a new exercise to support her learning. In practice, most systems rely on machine learning techniques to analyze large amounts of student interaction data and recommend suitable exercises. The main technical challenge is to design an optimal recommendation strategy (algorithm) that can recommend the best exercise for each student at the right time [4, 25].

Over the past years, many algorithms have been proposed for exercise recommendation from both educational psychology and data mining perspectives, such as collaborative filtering [32] and cognitive diagnosis [7]. Generally, exiting studies try to discover the weakness of students and then make decisions following a single strategy of recommending their non-mastered exercises (i.e., to address the weakness of students). Though it is reasonable, there are serious limitations. For example, if a student is estimated to be not good at the "Function" concept, the algorithm would tend to keep pushing her to practice "Function" exercises, including even the ones that she cannot deal with, which is non-optimal and ignores the long-term learning needs of students in practice [4].

Therefore, an ideal recommender system should be able to support an adaptive strategy of considering the following three additional objectives (instead of the above single one) for exercise recommendation: 1) *Review & Explore.* The primary goal of exercise recommendation is to help students acquire as much necessary knowledge as possible. So it requires us to help fix students' existing holes by making their non-mastered concepts more solid with timely reviews, and simultaneously provide them a chance to explore new knowledge even if their concept holes are not completely filled sometimes. For example, in Figure 1, it is desirable to still recommend another exercise with "Function" concept to Bob

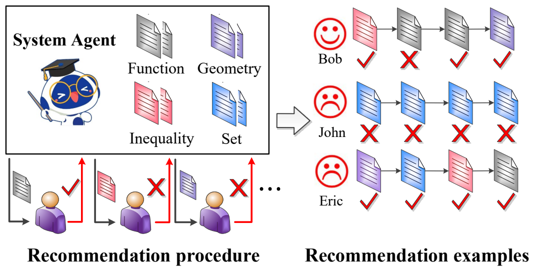


Figure 1: Example: An exercise recommendation procedure (left). Recommendation examples for three students (right).

at the second step since he answers a "Function" exercise wrong before, but a new "Geometry" exercise could be recommended next time. Hence we need to consider a trade-off between the *review* and *explore* [4] factors. 2) *Smoothness of difficulty level:* As students learn knowledge gradually, the difficulty levels of continuous recommendations should not vary dramatically [39]. For example, it is inappropriate if we push a primary student who has just learned "Addition and Subtraction" into a hard "Calculus" exercise. Therefore, it is necessary to smooth the exercises' difficulty levels for recommendations (In the following, we use *Smoothness* for simplicity). 3). *Engagement.* The engagement factor is of great importance to maintain the enthusiasm of students when they are exercising. As shown in Figure 1, neither John nor Eric is satisfied because they always find the recommended exercises too hard or easy for them. So it is desirable to adjust the recommendations so that some of them are challenging for students but others more like "gifts" in practice, which can help improve the engagement.

To the best of our knowledge, no previous work on exercise recommendation has attempted to optimize the three objectives discussed above simultaneously. In this paper, we address them in a principled way by proposing a novel *D*eep *R*einforcement learning framework for adaptively recommending *E*xercises to students, namely DRE. In this framework, we formalize the recommendation procedure as a Markov decision process (MDP) with several interactions between students and a recommender agent.

Due to the large number of exercises in our problem setting, it is infeasible to estimate all exercise transitions at each step as some Q-learning based approaches [4] do. We address this problem by leveraging deep learning methods to model the action-value function. Meanwhile, to fully capture the exercising information from students' interaction records, we propose two different novel Exercises Q-Networks (EQN), which generate recommendations based on different amount of historical information from the exercises done by a student. The first one is a straightforward yet effective *EQNM with Markov property*, where the next recommendations to students only depend on their current exercising performance. In contrast, the second is a more sophisticated *EQNR with Recurrent manner*, where we track all their exercising history to make decisions. Both architectures are model-free, and thus we do not need to calculate the state transitions in the whole exercise space.

In order to support adaptive recommendations with the above three domain-specific objectives, we propose three novel reward functions to capture and quantify the effects of them. As a result, DRE could update and optimize its recommendation strategy by interacting with students from the whole exercising trajectory.

We conduct extensive experiments to evaluate the proposed framework on two real-world datasets. Our experimental results show that the proposed DRE can effectively learn from the student interaction data to optimize multiple objectives in a single unified framework and adaptively recommend useful exercises to students.

Although our proposed framework was motivated by the recommendation applications in the education domain, the main ideas in our design of both the Exercise Q-Networks and the three reward functions are much more general, and can be potentially applicable to many other recommendation applications where some kind of feedback information from the users is available (equivalent to the performance of a student on an exercise). Indeed, the general goals behind all those three objectives that we attempt to model seem to be also desirable in many other recommendation applications.

2 RELATED WORK

In this section, we briefly review the related works as follows.

Cognitive Diagnosis. The primary goal of online education systems is to help students acquire necessary knowledge as much as possible, which requires to suggest learning materials that cohere with their knowledge states [4]. Therefore, one of the fundamental studies refers to cognitive diagnosis, aiming at discovering student mastery levels on knowledge concepts [5, 7, 16]. Existing cognitive diagnostic models could be grouped into two categories: unidimensional models and multidimensional models. Among them, Item Response Theory (IRT), as a typical unidimensional model, considered each student as a latent trait variable [7, 36]. In contrast, multidimensional models, such as *Deterministic Inputs, Noisy-And gate* (DINA), characterized each student by a binary vector which described whether or not she had mastered the knowledge concepts [6]. In addition, to capture the dynamics of earning, a group of Bayesian knowledge tracing [37] and deep learning tracing models [20, 21] try to update students' knowledge states. With the measured knowledge states of students, the recommender could select exercises for each student following fine-grained rules [25].

Recommender System. In recent years, recommender systems have achieved great success in a variety of domains, such as e-commerce [40], POI services [10] and social networks [33–35], due to their superiority of filtering confusing content. With massive student exercising data accumulated in online learning systems, several approaches leverage traditional algorithms for exercise recommendation including content-based filtering [25], collaborative filtering [32] and hybrid ones [30]. Specifically, content-based methods provide students with exercises owning the similar properties that they perform not well in the past. Collaborative filtering makes decisions for students driven by their neighbors with similar behaviors. One step further, hybrid methods take advantage of both, considering other factors such as ontology [30] and semantics [25]. Recently, deep learning techniques have advanced the performance [20, 27, 28] with deep structures for capturing complex student-exercise relationships. For example, Piech et al. [20] utilized recurrent neural networks to track the exercising dynamics. Su et al. [27] further considered the effects of text contents.

Reinforcement Learning in Education. Reinforcement learning is a kind of machine learning techniques, which has been examined in various fields, such as robotics [14], game [26] and recommender systems [40]. Generally, it holds a good ability of letting

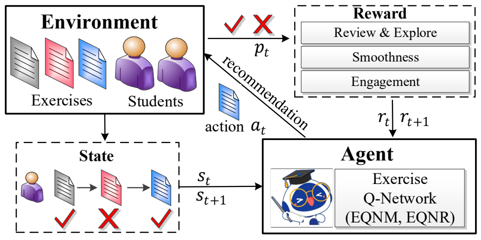


Figure 2: The proposed DRE framework.

software agents learn to take actions in an unfamiliar environment to maximize long-term reward. In educational psychology, state-of-the-art studies combine both cognitive diagnosis and reinforcement learning for exercise recommendation, such as multi-armed bandit (MAB) method [31] and Q-learning algorithm [29]. However, such approaches suffer from the problems as follows. First, they usually have high computational and representational complexity, where we have to estimate the transitions in all state-action spaces, and maintain an entire Q table memory for storage [3, 29]. Obviously, it is infeasible in real-world online education systems since they usually contain large number of exercise candidates to be recommended. Second, they only exploit students' current grades for estimating their states while ignoring the effects of more beneficial information including exercise content and knowledge concept.

Our proposed DRE mainly has the following advantages compared with the previous studies. First, DRE includes the flexible Exercise Q-Network, a function approximator, to select exercises at each step, alleviating the high complexity of computation and storage. Second, DRE learns students' states with the exercise contents, knowledge concepts as well as performance scores and proposes different novel strategies for recommendations from their long-term exercising. Third, DRE, to the best of our knowledge, is the first comprehensive attempt to simultaneously optimize three educational domain-specific objectives including *Review & Explore*, *Smoothness* and *Engagement*, supporting adaptive exercise recommendations.

3 PROBLEM AND FRAMEWORK OVERVIEW

In this section, we first formalize the exercise recommendation problem and introduce the overview of DRE framework.

3.1 Problem Statement

In an online education system, suppose there are $|U|$ students and $|E|$ exercises. We record the exercising process of a certain student $u = \{(e_1, p_1), (e_2, p_2), \cdots, (e_T, p_T)\}, u \in U$, where $e_t \in E$ represents the exercise that student u practices at her time step t, and p_t denotes the corresponding performance. Generally, if she answers exercise e_t right, p_t equals to 1, otherwise p_t equals to 0. For a certain exercise $e \in E$, we describe it with a triplet as $e = \{c, k, d\}$. Specifically, the element c represents its text content as a word sequence $e = \{w_1, w_2, \ldots, w_M\}$. $k \in K$ describes its knowledge concept (e.g., *Function*) coming from all K concepts. And d means its difficulty factor. In this paper, we define the difficulty d as the exercise e's error rate calculated from logged data, i.e., the percentage of students who answer exercise e wrong [11, 12].

In this paper, we formalize the procedure of exercise recommendation as a Markov Decision Process (MDP) in which the recommender agent interacts with students in a sequential decision making procedure. Our formulation is similar to the most recent one [29], however with two important differences: (1) we incorporate richer information about each exercise triple of $e = \{c, k, d\}$ into the state representation; (2) we design a flexible reward function reflecting multi-objective goals. More formally, we define our MDP tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$ as follows:

- *States* $\mathcal{S}$: $\mathcal{S}$ is the state space which models students. At time t, the state $s_t \in \mathcal{S}$ is defined as the preceding exercising history of a student. The triple information of each exercise $e = \{c, k, d\}$ is also considered.
- *Actions* $\mathcal{A}$: $\mathcal{A}$ is the action space, containing all exercises in the system. Taking an action $a_t \in \mathcal{A}$ based on state s_t is defined as recommending an exercise e_{t+1} to the student.
- *Reward* $\mathcal{R}$: $\mathcal{R}(\mathcal{S}, \mathcal{A})$ is a reward function. After the agent takes an action a_t at state s_t, the student learns and answers it. Then the agent receives reward $r(s_t, a_t)$ according to her performance p_t (feedback), where we design it with multiple objectives. We will discuss it in detail in Section 4.3.
- *Transitions* $\mathcal{T}$: Transition $\mathcal{T}(\mathcal{S}, \mathcal{A})$ is a function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which maps a state s_t into a new state s_{t+1} after the agent takes action a_t.

With the above formulation, our goal is to find an optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ of recommending exercises to students, in the way that maximizes the multi-objective rewards for recommendations.

3.2 Framework Overview

Figure 2 illustrates the overview of DRE framework with the MDP formulation. Our environment is made up with student pool and exercise pool. Our agent is designed with the flexible Exercise Q-Networks (EQN). At time step t, the agent receives a student's state s_t with her exercising records, and then selects an action a_t, i.e., recommending an exercise from the exercise pool. Then the student learns and answers it. One time step later, the agent receives reward r_t based on her performance p_t. Specifically, the reward is designed with a multi-objective one considering the effects of three domain-unique characteristics including *Review & Explore*, *Smoothness* and *Engagement*. After that, DRE moves to a new state $s_{t+1} = \mathcal{T}(s_t, a_t)$.

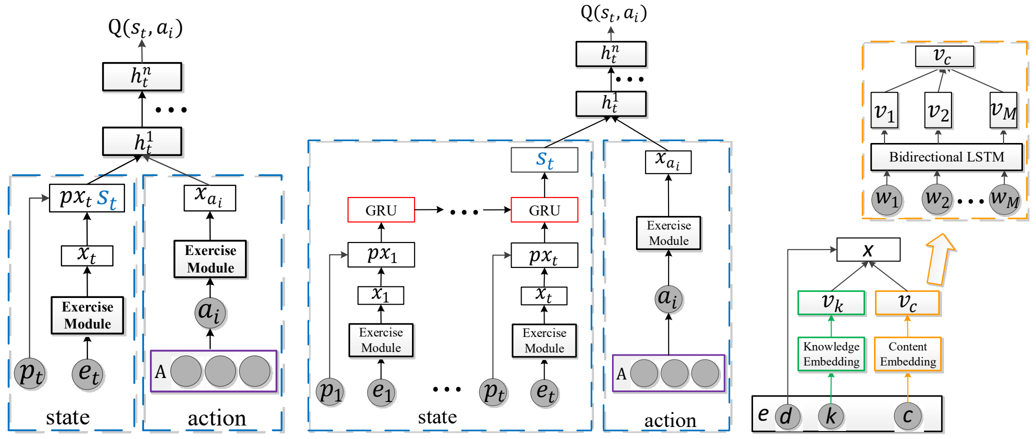
Specifically, in DRE framework, we need to deal with three main challenges: (1) how to effectively generate recommendations from exercise pool (with EQN in Section 4.2); (2) how to quantify the multiple domain-objective rewards for making adaptive recommendations (Section 4.3); (3) how to train DRE to update the recommendation policy π with offline logged exercising data (Section 4.4).

4 DRE FRAMEWORK

In this section, we introduce the details of DRE framework including the optimization objective, Exercise Q-Network and multi-objective reward design. After that, we present the training procedure.

4.1 Optimization Objective

To seek the optimal policy π, standard reinforcement learning [18] finds the optimal action-value function $Q^*(s, a)$ of taking action a at state s to maximize the expected future rewards. Here, the future rewards R_t of state-action pair (s, a) are discounted by a factor $\gamma \in (0, 1)$ per step, i.e., $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where $r_{t'}$ is the reward at step t'. Therefore, the optimal function $Q^*(s, a)$ will satisfy the



(a) EQNM with Markov property (b) EQNR with Recurrent manner (c) Exercise Module

Figure 3: Two Exercise Q-Networks, where shaded and unshaded symbols denote observed and latent variables, respectively.

Bellman equation [2] as follows:

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a')|s,a]. \tag{1}$$

In Eq. (1), selecting the optimal action a' requires to compute the Q-values for all $a' \in \mathcal{A}$ separately. However, it is infeasible in online education systems due to the following two main reasons. First, there are extremely large volume of exercises to be recommended, which brings us a problem of estimating and storing all state-action pairs. Second, since a student usually practices very few exercises compared with all exercise space, this sparse observation causes a difficulty to update the Q-values of the state-action pairs and calculate the state transitions if she does not practice the corresponding exercises.

To alleviate these two problems, following [18], DRE framework leverages the deep reinforcement learning solution, which utilizes a non-linear function approximator θ to estimate the action-value function, i.e., $Q^*(s,a) \approx Q(s,a;\theta)$ in Eq. (1), instead of maintaining the entire Q table. In this paper, we design two novel Exercise Q-Networks to implement the approximator θ following different mechanisms (details will be discussed in Section 4.2). To estimate this network approximator, we could minimize the following loss function $L_t(\theta_t)$ at step t as:

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'}[(y - Q(s,a;\theta_t))^2], \tag{2}$$

where $y = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q(s',a';\theta_{t'})|s,a]$ is the target for current iteration t. The parameters $\theta_{t'}$ represent the previous target network, which are fixed when optimizing the $L_t(\theta_t)$. As a result, the derivatives of loss function Eq. (2) can be defined as:

$$\begin{aligned}\nabla_{\theta_t} L_t(\theta_t) =& \mathbb{E}_{s,a,r,s'}[(r + \gamma \max_{a'} Q(s',a';\theta_{t'}) \\ &- Q(s,a;\theta_t))\nabla_{\theta_t} Q(s,a;\theta_t)].\end{aligned} \tag{3}$$

As [18, 40] suggested, directly computing the full expectations of the gradient in Eq. (3) can be of high computational complexity. In this paper, we use Adam [13] method to optimize the loss function.

4.2 Exercise Q-Network

Now, we deal with the first challenge of implementing the function approximator θ in Eq. (2). That is to design an appropriate network, which we call Exercise Q-Network (EQN), with the goal to estimate the action Q-value $Q(s,a)$ of taking an action a at state s. Indeed, a natural choice is the standard DQN [18], which utilizes several fully connected layers for the estimation. However, such simple structure cannot fully capture the information from students' exercising records to represent the state s_t, and usually fails to explore the effects from their historical exercises including the contents, concepts and performance scores. Therefore, as shown in Figure 3, in this paper, we propose two implementations of EQN following two mechanisms. The first one is *EQNM with Markov property*, which just observes the latest one-time exercising record of the student. The second one is *EQNR with Recurrent manner*, tracking the long-term effects of her whole history. We will first introduce the essential *Exercise Module* in both EQNs, followed by two EQNs.

4.2.1 Exercise Module. The goal of *Exercise Module* is to learn the semantic encoding x of exercise e with its triplet input $\{c, k, d\}$ at each time. Naturally, as shown in Figure 3(c), we can first encode its concept k (*Knowledge Embedding*) and content c (*Content Embedding*) with separate components and then combine both as well as its difficulty feature d into an overall vectorial representation.

For *Knowledge Embedding*, given the concept k, we first set it to be a one-hot encoding $k \in \{0,1\}^K$. Since this intuitive one-hot representation is too sparse for modeling [9], we introduce the embedding matrix $\mathbf{W_k}$ to transfer it into a low-dimensional vector $v_k \in \mathbb{R}^{d_k}$ with continuous values as: $v_k = \mathbf{W_k}^{\mathrm{T}} k$. Note that v_k reflects the semantics of exercise e in the knowledge space.

For *Content Embedding*, given the text content sequence with M words $c = \{w_1, w_2, \ldots, w_M\}$, we first preliminarily take *Word2vec* to transform each word w_i into a d_0-dimensional pre-trained word embedding [17]. Then, we design a bidirectional LSTM, an improved *Long Short-Term Memory* (LSTM), to learn contextual encoding v_i of each word w_i. Here, we adopt bidirectional LSTM because it can make the most of contextual word information of exercise sentence from both forward and backward directions [21]. Specifically, the bidirectional LSTM could be formulated as:

$$\begin{aligned}\overrightarrow{v}_i = LSTM(w_i, \overrightarrow{v}_{i-1}; \theta_{\overrightarrow{v}}), \quad \overleftarrow{v}_i = LSTM(w_i, \overleftarrow{v}_{i-1}; \theta_{\overleftarrow{v}}), \\ v_i = \overrightarrow{v}_i \oplus \overleftarrow{v}_i,\end{aligned} \tag{4}$$

where $v_i \in \mathbb{R}^{d_v}$ is the context of word w_i, which concatenates ($\oplus$) both its forward context $\overrightarrow{v}_i$ and backward context $\overleftarrow{v}_i$. After that, to obtain the contextual representation of exercise e, we exploit the element-wise max pooling operation to merge M word contexts into a global embedding as $v_c = \text{maxpool}(v_1, v_2, \ldots, v_M)$. Hence, v_c reflects the semantics of exercise e in the context space.

One step further, the overall semantic encoding x of exercise e could be concatenated with both its knowledge and context semantics, i.e., v_k and v_c, as well as its difficulty factor d as: $x = v_k \oplus v_c \oplus d$.

4.2.2 EQNM with Markov property. EQNM holds a straightforward mechanism following the Markov property. It is a commonly made assumption, defining that the next state depends only on the current state and not on the sequences that precede it [22]. Given this theory, EQNM just observes the student's current exercising record to take the action. Formally, we redefine the state in EQNM as:

- *State s*: The state $s_t = (e_t, p_t)$ is defined as a student's exercising record at time step t, consisting of the exercise e_t that the student practices as well as her performance p_t.

Figure 3(a) illustrates the network architecture of EQNM. Given the state $s_t = (e_t, p_t)$, EQNM needs to learn the state embedding. Since students getting right answer (i.e., score 1) or wrong answer (i.e., score 0) to the same exercise actually reflect their different states [21], it is necessary to distinguish these effects for state embedding. Formally, EQNM first leverages *Exercise Module* to extract the exercise encoding x_t, and then concatenates it with the performance p_t to learn the combined exercising encoding px_t as:

$$px_t = \begin{cases} [x_t \oplus \mathbf{0}] & \text{if} \quad p_t = 1, \\ [\mathbf{0} \oplus x_t] & \text{if} \quad p_t = 0, \end{cases} \tag{5}$$

where $\mathbf{0} = (0, 0, \ldots, 0)$ is a performance feature vector with the same dimensions of x_t. As a result, in EQNM, we naturally consider the exercising embedding px_t as the current state, i.e., $s_t = px_t$.

After that, based on state s_t, EQNM outputs the action Q-value $Q(s_t, a_i)$ of each exercise candidate $a_i \in \mathcal{A}$ in action space, with several (n) fully-connected layers to learn the high-level state-action features. On the top, we output the Q-value by $Q(s_t, a_i) = \frac{1}{\exp(-h_t^n)}$.

4.2.3 EQNR with Recurrent manner. Although EQNM provides a straightforward yet effective way for estimating the action Q-value $Q(s, a)$, in real-world practice, the learning process of a student is not static but evolves over time in which some more previous exercising performance of her could also benefit recommendations. Therefore, considering this long-term dynamic nature, we propose a more sophisticated *EQNR with Recurrent manner*, which takes the action based on the student's whole exercising trajectories. Formally, we redefine the state in EQNR as:

- *State s*: The state $s_t = \{(e_1, p_1), \cdots, (e_t, p_t)\}$ contains all the student's historical exercising records from time step 1 to t.

Figure 3(b) illustrates the architecture of EQNR. In comparison with EQNM, EQNR introduces a GRU network, i.e., $GRU(\cdot; \theta_g)$, to track the student's exercising history, where the long-term dependency of the state s_t could be well captured. Here, we choose GRU rather than LSTM in our implementation as it is computational efficient and has comparable performance with LSTM [15]. Formally, GRU updates the hidden exercising state l_j as:

$$l_j = GRU(px_j, l_{j-1}; \theta_g). \tag{6}$$

where px_j at each time is calculated by Eq. (5). Then we consider state s_t as the output of final hidden exercising state l_t, i.e., $s_t = l_t$.

After that, EQNR estimates the Q-value $Q(s_t, a_i)$ of each exercise candidate with the same operation in EQNM.

4.3 Multi-Objective Rewards

Now we discuss the second crucial issue of how to design our reward $r(s_t, a_t)$, which plays an important role to help DRE learn the optimal recommendation policy π. As mentioned before, traditional recommender systems (e.g., collaborative filtering [32]), usually follow a fixed strategy of recommending non-mastered exercises to the student. However, such single-objective recommendations cannot satisfy the needs of students in practice [4]. In this paper, we incorporate three major domain-specific factors into the reward estimation including *Review & Explore*, *Smoothness* and *Engagement*, supporting the adaptive exercise recommendations. Along this line, we design our reward function $\mathcal{R}(\mathcal{S}, \mathcal{A})$ in a multi-objective way.

Review & Explore. To help students acquire necessary knowledge more effectively, a recommender agent should balance *Review* and *Explore* factors [4]. The *Review* factor requires to help students review what they learned not well to fix their concept holes. This implies that a punishment ($\beta_1 < 0$) should be given if the agent recommends an exercise with different concept when she answers one wrong just before. Comparatively, the *Explore* factor suggests that we should seek diverse concepts to help them acquire more knowledge even if their holes are not completely filled. Guiding with this factor, we give a stimulation ($\beta_2 > 0$) if the agent accesses a new concept for recommendation. Formally, we design the first reward trading-off both factors as:

$$r_1 = \begin{cases} \beta_1 & \text{if} \quad p_t = 0 \quad \text{and} \quad k_{t+1} \cap k_t = \emptyset, \\ \beta_2 & \text{if} \quad k_{t+1} \setminus \{k_1 \cup k_2 \cup \cdots \cup k_t\} \neq \emptyset, \\ 0 & \text{else}. \end{cases} \tag{7}$$

Please note that r_1 is a flexible reward where both β_1 and β_2 could be set in different scenarios. For example, we can set β_2 with lager values if we hope the agent focus more on exploring new knowledge concepts in practice.

Smoothness. As discussed before, students usually learn knowledge gradually, so the difficulty levels of continuous recommendations should not vary dramatically [39]. This is an important problem. However, to the best of our knowledge, none of existing works consider it for exercise recommendation. Though there is no standard solution to it, yet intuitively, we could make it negatively correlated with the difference between the difficulty levels of the two exercises. Along this line, there are many potential ways for the *Smoothness* reward design. Here, for simplicity, we apply a commonly used squared loss function to smooth two recommendations at continuous time steps from the difficulty view as:

$$r_2 = \mathcal{L}(d_{t+1}, d_t) = -(d_{t+1} - d_t)^2, \tag{8}$$

where $\mathcal{L}(\cdot, \cdot)$ is the negative squared loss. Larger $\mathcal{L}(\cdot, \cdot)$ indicates the closer difficulties of two exercises, leading a stronger stimulation.

Engagement. It is of great importance to keep students always engaged in the exercising, However, this is highly challenging to

Algorithm 1: DRE Learning with Off-Policy Training

```
1  Initialize replay memory $\mathcal{D}$ with capacity $Z$;
2  Initialize action-value function $Q$ with random weights.;
3  for $u = 1, 2, \cdots, |U|$ do
4      Randomly initialize state $s_0$;
5      for $t = 1, 2, \cdots, T$ do
6          Observe state $s_t = (e_t, p_t)$ in EQNM or
             $s_t = \{(e_1, p_1), \cdots, (e_t, p_t)\}$ in EQNR;
7          Execute action $a_t$ $(e_{t+1})$ from off-policy $\pi_o(s_t)$;
8          Compute reward $r_t$ according to $p_{t+1}$ by Eq. (10);
9          Set state $s_{t+1} = (e_{t+1}, p_{t+1})$ in EQNM or
             $s_{t+1} = \{(e_1, p_1), \cdots, (e_t, p_t), (e_{t+1}, p_{t+1})\}$ in EQNR;
10         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$;
11         Sample minibatch of transition $(s, a, r, s')$ from $\mathcal{D}$;
12         $y = \begin{cases} r & \text{terminal } s' \\ r + \gamma \max_{a'}(Q(s', a'); \theta) & \text{non-terminal } s' \end{cases}$;
13         Minimize $(y - Q(s, a); \theta)^2$ by Eq. (3);
14     end
15 end
```

model and none of prior studies has noticed that. To seek the solution, we find that in Figure 1, either John or Eric feels bored with the recommendations because they always feel incapable or not challenged about the exercises. Therefore, there is an appropriate way that we could adjust the recommendations that makes some of them are challenging but others seem more like "gifts" for students. Intuitively, if a student always performs well recently, we need to increase the difficulty levels of recommendations, and vice versa. To achieve this goal, we design the *Engagement* reward as:

$$r_3 = 1 - |g - \varphi(u, N)|, \quad \varphi(u, N) = \frac{1}{N} \sum_{i=t-N}^{t} p_i, \tag{9}$$

where we call $g \in [0, 1]$ the "learning goal" factor. We also set a window with size N to let DRE recall the student most recent N exercising records and calculate her average performance $\varphi(u, N)$. The basic idea of Eq (9) is that we hope the recent average performance of the student $\varphi(u, N)$ approaches the learning goal g. So if the gap between g and $\varphi(u, N)$ is larger, our agent will receive lower reward, then it can adaptively adjust recommendations.

It is worth mentioning that the learning goal g is flexible in practice, which can be set by the instructor or students themselves. For example, if someone expects more challenging exercising experience, she can set g with a lower value.

As a result, we merge the above three reward with α_1, α_2, α_3 balance coefficients as:

$$r = \alpha_1 \times r_1 + \alpha_2 \times r_2 + \alpha_3 \times r_3, \quad \{\alpha_1, \alpha_2, \alpha_3\} \in [0, 1]. \tag{10}$$

Eq. (10) also reflects a flexible way to merge multi-objective rewards in DRE. That is, we can set them with different goals in real world applications. For example, we can set α_1 with a larger value if we hope DRE focuses more on the first *Review & Explore* objective. Specifically, if we respectively set $\alpha_1, \alpha_2, \alpha_3$ as 0, 0, 1 in Eq. (10), and g as 0 in Eq (9), DRE will follow the common strategy of recommending non-mastered exercises as many existing works do. We will discuss more detailed settings in the experiments.

In summary, our proposed DRE framework mainly has the following advantages. First, DRE holds a flexible Exercise Q-network to estimate the state-action Q-values, instead of directly calculating and maintaining the entire Q table in the whole space, which is feasible in education systems. Second, DRE deeply captures the effects from students' exercising historical trajectories for recommendations following two mechanisms. Third, with the multi-objective rewards, DRE takes various domain-specific characteristics for adaptively recommending exercises to students during the learning process, rather than just following the single "non-mastered" suggestion, Last, our framework is model-free. It can solve the reinforcement learning task directly using samples from the student records.

4.4 Framework Learning

In reinforcement learning, there is a problem of data inefficiencies by collecting large quantities of training data. To deal with this problem, classical applications usually resort to the solution of self-play or simulation [14, 26]. Nevertheless, for exercise recommendations, it is hard to simulate real data with our policy due to its complex dynamics coherent. Thus, we cannot receive rewards in unexplored space, since observing rewards requires giving a real exercise recommendation to a real student.

To solve this problem, we take an off-policy approach [40], making full use of students' offline logs from other agent policy ($\pi_o(s_t)$) to update our recommendation policy ($\pi(s_t)$). We present our off-policy learning algorithm in Algorithm 1. Particularly, we introduce the following widely used techniques in the training procedure. First, we adopt *experience reply* [24] to store the agent's latest Z experiences (line 10). Then, we implement two separate networks [18], i.e., evaluation network and target network, to avoid the divergence of parameters when learning (line 13).

5 EXPERIMENT

In this section, we conduct extensive experiments to validate the effectiveness of DRE framework.

5.1 Experimental Dataset

We used two real world datasets, namely MATH and PROGRAM. The MATH dataset, supplied by iFLYTEK Co., Ltd, was collected from their online education system called Zhixue (zhixue.com), which helps high school students self-exercising on many subjects. We collected the mathematical data records. In the system, when an exercise was posted to a student, she answered it and got the score. Then she could get the solution if she accessed the "hint" or stepped into the next. In this scenario, we just recorded her first-attempt responses and the timestamps for fairness. To make the reliability of experimental results, we filtered the students that did less than 10 exercises and the exercises that no students had done. All exercises belonged to 37 concepts, such as "Function" and "Geometry".

The PROGRAM data were crawled from PKU JudgeOnline platform (poj.org), enabling students self-training about programming (we omit both system names for submission due to the anonymity principle). In the OJ system, students were allowed to resubmit their answers until they passed the assessment. Thus we could not directly take the final "pass" or "failed" status as their performance

Table 1: The statistics of the datasets.

Dataset	Num. Students	Num. Exercises	Num. Conceps	Num. records	Avg. records per student
MATH	52,010	2,464	37	1,272,264	24.5
PROGRAM	40,013	2,900	18	3,455,067	86.3

because students with different attempt times performed inconsistently. To deal with this phenomenon, we developed a function $P(y) = \exp(\frac{1-y}{4})$ to transfer their submission times y into the performance score p, and however if the student failed at last, p=0. Correspondingly, we left their latest submission timestamps for the exercising time. Moreover, without loss of generality, we kept the 18 main knowledge concepts, such as "Number Theory" and "Data Structure". We also filtered the students that did less than 20 exercises. More statistics of our datasets are presented in Table 1[1].

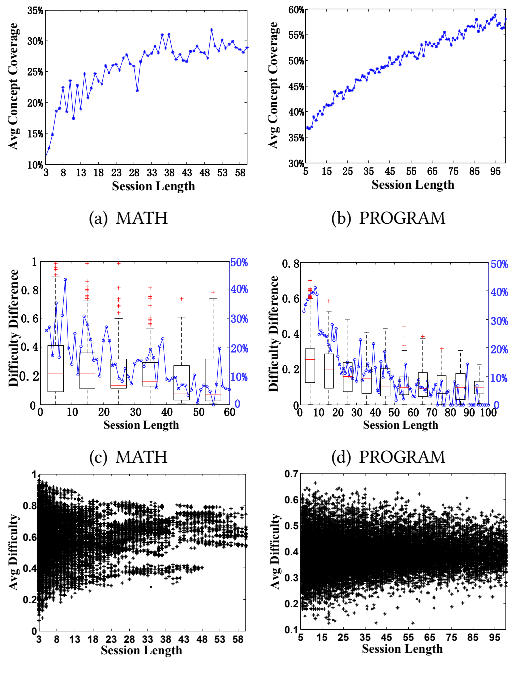
Dataset Analysis. We deeply analyzed both datasets in Figure 4. Here, we first aggregated their exercising logs into different sessions. For a certain student, if the interval timestamps of her two continuous exercising records lasted more than 24 (10) hours in MATH (PROGRAM), we split them into two sessions, and thus her whole records could be aggregated into different sessions. Obviously, more exercising submissions in the session (i.e., longer sessions) reflect that she is willing to spend more time for learning.

Specifically, first, Figure 4(a) and Figure 4(b) show the average concept coverage of all exercises in sessions with different lengths. Obviously, longer sessions have larger concept coverage, which means the student would learn more knowledge. This observation guides us to explore new concepts for recommendations if we hope students keep learning. Second, Figure 4(c) and Figure 4(d) calculate the difficulty difference between two continuous exercises in each session. Here, the box charts depict their distribution in each group (we aggregated the groups with every 10 session lengths, i.e., [0, 10), [10, 20), $\cdots$), and the line chart shows the proportion of them greater than 0.4 in sessions with different length (if the difficulty difference of two exercises is greater than 0.4, we define that this change is dramatic). From the figure, sessions with shorter length contain more samples with larger difficulty differences, and simultaneously, longer exercising sessions have more smooth exercises that the student practices. Last, Figure 4(e) and Figure 4(f) illustrate the scatter analysis of the average difficulty of all exercises in each session. We find that longer sessions have exercises with medium difficulty on average, which means students would be engaged in the exercising by choosing both hard and easy exercises. In summary, all the above analyses could support the rationality of exploring our proposed three objectives including *Review & Explore*, *Smoothness* and *Engagement* for exercise recommendations.

5.2 Experimental Setup

Exercise Q-Network. In EQNs, we initialized word embedding in *Exercise Module*. We pretrained each word in exercises into a embedding vector with the 50 dimensions (i.e., $d_0 = 50$) by word2vec [17]. For the network initialization, we set the dimension d_k=10 in Knowledge Embedding, d_v=100 in Content Embedding, d_l=100 in EQNR, respectively. Moreover, in both EQNM and EQNR, we leveraged 2 (i.e., n=2) fully connected layers for Q-value estimation (Figure 3).

[1]The data is available at: https://base.ustc.edu.cn/data/DRE/



(a) MATH (b) PROGRAM (c) MATH (d) PROGRAM (e) MATH (f) PROGRAM

Figure 4: Exercising record analysis in both datasets.

Training Setting. To set up training process, we initialized all network parameters in DRE following [19]. Each parameter was sampled from $U(-\sqrt{6/(n_{in}+n_{out})}, \sqrt{6/(n_{in}+n_{out})})$ as their initialized values, where n_{in} and n_{out} denoted the numbers of neurons feeding in and fed out, respectively. Besides, we set the discounted factor $\gamma = 0.9$ in Eq. (1). We set the capacity of replay memory Z=500, and the minibatch as 32 in Algorithm 1.

Reward Setting. There are several reward hyperparameters need to be set in the experiments, i.e., punishment β_1 and stimulation β_2 in Eq. (7), window size N and learning goal g in Eq. (9), and balance coefficients $\alpha_1, \alpha_2, \alpha_3$ in Eq. (10). As mentioned before, all these hyperparameter settings are flexible for different scenarios. Thus, we will discuss them later in the experiments.

For simplicity, we call our two DRE based models with two Exercise Q-Netoworks (i.e., EQNM and EQNR), as DREM and DRER, respectively. All experiments were conducted on a Linux server with four 2.0GHz Intel Xeon E5-2620 CPUs and a Tesla K20m GPU.

5.3 Offline Evaluation

There are two typical scenarios of exercise recommendation in real world applications, i.e., point-wise recommendation, and sequence-wise recommendation, respectively. Specifically, in the point-wise recommendation, we need to provide an exercise list for each student at a particular time. All exercises in the list are ranked by algorithms at the same time. In the sequence-wise scenario, we have to recommend exercises to a specific student step by step, where at each step, we suggest the best exercise, receive the feedback and then make the following decision in next round.

In the offline evaluation, we conducted experiments with the logged exercising records. The logged data were static and only contained certain pairs of student-exercise performance that had

Table 2: The overall accuracy results of exercise recommendation in offline evaluation.

(a) **MATH**

Methods	NDCG@10	NDCG@15	MAP@10	MAP@15	F1@10	F1@15
IRT	0.5065	0.6235	0.3373	0.4463	0.2100	0.3464
PMF	0.4900	0.5986	0.3155	0.4163	0.2016	0.3347
FM	0.5123	0.6279	0.3419	0.4507	0.2123	0.3489
DKT	0.5587	0.7033	0.3959	0.5486	0.2797	0.4634
DKVMN	0.5657	0.7112	0.4021	0.5581	0.2895	0.4747
DQN	0.5031	0.7001	0.3191	0.5296	0.2912	0.5178
DREM	**0.6114**	0.7773	**0.4355**	0.6353	0.3559	0.6033
DRER	0.6129	**0.7813**	0.4337	**0.6435**	**0.3676**	**0.6099**

(b) **PROGRAM**

Methods	NDCG@10	NDCG@15	MAP@10	MAP@15	F1@10	F1@15
IRT	0.3369	0.4231	0.1852	0.2430	0.0879	0.1530
PMF	0.3330	0.4152	0.1810	0.2336	0.0842	0.1467
FM	0.3664	0.4456	0.2081	0.2617	0.0921	0.1567
DKT	0.3893	0.4924	0.2361	0.3197	0.1451	0.2445
DKVMN	0.3853	0.4889	0.2351	0.3226	0.1555	0.2620
DQN	0.3422	0.4901	0.1851	0.3095	0.1781	0.3266
DREM	0.4446	0.5638	0.2753	0.3834	0.1683	0.3325
DRER	**0.4538**	**0.5907**	**0.2802**	**0.4059**	**0.2091**	**0.3655**

been recorded. As a result, we cannot make sequential recommendations since we lack the real-time feedback from students with the offline data. Hence, it is hard to dynamically track the benefits of domain-specific rewards. Therefore, in this scenario, we focus on point-wise recommendation to demonstrate the accuracy of DRE.

To setup, we partitioned each student's exercising sequence into training/test sets. Specifically, for a certain student, we used her beginning 70% records as training sets, and the remaining 30% for testing. please note that in the test sets, we just observed students' real performance on the exercises that they had practiced. Thus we only used these exercises as ground truth for evaluation. For the offline test, given a test student, DREM and DRER rank recommendation list according to the estimated Q-values from high ones to the low, based on her state at the last step in the training sets.

To evaluate the recommendation accuracy of DREM and DRER, we targeted at a special case of recommending non-mastered exercises to students, which is the most common scenario in practice. Our goal was to rank the exercises that students answered wrong at the top of recommendation list. Along this line, we adapted our DREM and DRER, i.e., we set the reward settings as: r (α_1=0, α_2=0, α_3=1) (Eq. (10)); r_3 (g=0, N=5) (Eq. (9)). The baselines as follows:

- *IRT*: Item Response Theory [7] is a popular cognitive diagnostic model to discover their knowledge levels for ranking.
- *PMF*: Probabilistic Matrix Factorization [32] is a typical model-based collaborative filtering method, using a factorization model that projects students and exercises into latent factors.
- *FM*: Factorization Machine [23] combines the advantage of both support vector machines and matrix factorization, where higher-order interactions of features are considered.
- *DKT*: Deep Knowledge Tracing [20] is a recent deep learning method that leverages recurrent neural network to model student exercising process to rank the exercises. We implemented DKT with long short-term memory (LSTM).
- *DKVMN*: Dynamic Key-Value Memory Network [38] is a state-of-the-art model to trace students' states with a memory network. It contains a *key* matrix to store concept embedding and a *value* matrix for state modeling. Then, it can rank the results for students based on their states.
- *DQN*: We use the traditional Deep Q-network [18], replacing our proposed Exercise Q-network, for recommendations. We represent its states as the one-hot embedding with student's current exercising record ($s_t = (e_t, p_t)$). DQN also incorporates the same reward settings with DRE.

Table 2 reports the recommendation accuracy result in terms of the commonly used top@k ranking metrics [40] including NDCG, MAP and F1. We set k={10, 15} for each metric (we repeated all the experiments 5 times and averaged the results). There are some observations. First, DRER and DREM perform the best on both datasets, which indicates that DRE can optimize the recommendation policy by interacting with students so that makes accurate recommendations. Second, as our DRER and DREM incorporate the sophisticated Exercise Q-Networks (EQNR and EQNM) for state embedding, they perform better than simple DQN. This observation shows that our proposed EQNs could well capture the state presentations of students' exercising information including exercise contents, knowledge concepts and performance scores for recommendations. Third, DRER outperforms DREM, which demonstrates that it is effective to incorporate GRU to track the long-term dependency of exercising records for exercise recommendation. Moreover, we notice that the improvement of DRER on MATH dataset is not significant compared with it on PROGRAM. This proves that DREM, although with simple structure, can also guarantee the comparable recommendation performance when facing the sparse record data (we can see from Table 1 that the average exercising lengths of students in MATH (24.5) are much shorter compared with PROGRAM (86.3)). Last, traditional methods (IRT, PMF, FM) do not perform as well as all deep learning based models (DKT, DKVMN) and the reinforcement learning based models (DQN, DREM, DRER). We guess a possible reason is that they ignores the temporal nature of the student's exercising history. In summary, we can conclude that DRE is effective in making accurate exercise recommendations by taking full advantages of students' exercising trajectories.

5.4 Online Evaluation

In the online evaluation, we focused on the sequence-wise recommendation where we recommended exercises to students step by step. Here, we tested DRE in a simulated environment where it interacted with the student simulator and received real-time rewards calculated by the predicted performance for recommendations. In this scenario, we focused on evaluating the effectiveness of all rewards, i.e., *Review & Explore*, *Smoothness* and *Engagement*.

To set up the online test, we should first prepare a simulator that could accurately predict the performance given an exercise. Here, we chose EERNN as our simulator due to its prediction superiority in the literature [27] (Please note that comparing the performance of simulators is not the major concern of this work). Then, we

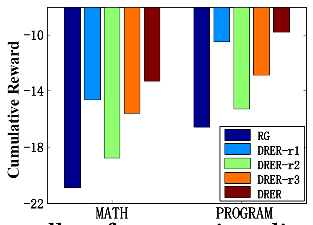


Figure 5: Overall performance in online evaluation.

divided the students with 50%/50% into two subsets on each dataset. Specifically, we used one subset to train EERNN simulator and the other for training DRE. Moreover, for training simulator, we also leveraged the last 10% records for validating its predictive performance. The results showed that EERNN simulator had over 80% prediction accuracy, which were consistent to results in [27]. Thus, EERNN simulator could support our online evaluation.

Recall that there are few existing approaches considering the domain-specific objectives for exercise recommendations. We designed a simple Random Greedy method and incorporate some variants of DRER as baselines. (We just used DRER to better illustrate the reward effectiveness). The details of all methods are:

- *RG*: Random Greedy follows the simple strategy that randomly recommends exercises at each step.
- *DRER-r_1*: The variant only contains the *Review & Explore* reward: r (α_1=1, α_2=0, α_3=0) (Eq. (10)), r_1 (β_1=-1, β_2=1) (Eq. (7)).
- *DRER-r_2*: The variant only contains the *Smoothness* reward: r (α_1=0, α_2=1, α_3=0) (Eq. (10)).
- *DRER-r_3*: The variant only contains the *Engagement* reward: r (α_1=0, α_2=0, α_3=1) (Eq. (10)), r_3 (g=0.5, N=5) (Eq. (9)).
- *DRER*: DRER contains all three rewards: r (α_1=1, α_2=1, α_3=1) (Eq. (10)), r_1 (β_1=-1, β_2=1) (Eq. (7)), r_3 (g=0.5, N=5) (Eq. (9)).

5.4.1 *Online performance.*

We set 20 sequential recommendations for each method and calculated their cumulative rewards as the metric for online evaluation. The overall results are shown in Figure 5. We can conclude as follows. First, DRER performs the best, followed by the variants. This evidence suggests that all three domain-specific rewards can benefit exercise recommendation and DRE framework can find optimal strategy by considering them simultaneously. Then, among three variants, DRER-r_1 performs the best, indicating that the *Review & Explore* reward plays more important role than others for exercise recommendation.

5.4.2 *Reward effectiveness.*

Now, we track the dynamics of three domain-specific rewards, i.e., *Review & Explore*, *Smoothness* and *Engagement*, to observe how they work in DRER.

Review & Explore. The *Review & Explore* reward helps DRE framework make a trade-off between reviewing existing knowledge and exploring new concepts. In Figure 6, we compare this performance of DRER with different stimulation values β_2 in Eq. (7) by calculating the cumulative concept coverage at each step during recommendation process. We can see: (1) As the recommendation process progresses, the concept coverage value of all DRER models increase; (2) DRER with larger β_2 has faster coverage growth speed than those with smaller ones. Therefore, we can conclude that with the *Review & Explore* reward, DRE could well seek multiple concepts for recommendations. Moreover, DRE would have more impacts on exploration if we set the stimulation β_2 with larger values.

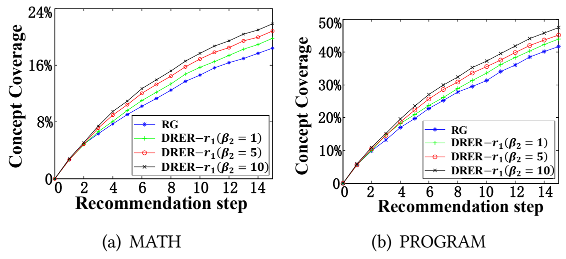


Figure 6: Results of Review & Explore reward.

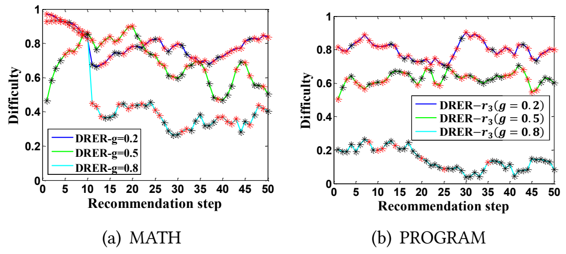


Figure 7: Results of Smoothness vs. Engagement rewards.

Smoothness vs. Engagement. DRE framework has two advantages of incorporating both *Smoothness* and *Engagement* rewards. Specifically, *Smoothness* factor controls the difficulty level of recommendations, and *Engagement* lets students self-set different learning goals, i.e., g in Eq. (9). Figure 7 visualizes two recommendation examples with these two rewards based on the exercises in both datasets. To set up, we first set different learning goals g={0.2, 0.5, 0.8} for three simulated students using DRER model. Then we tracked the difficulty levels of each exercise during 50 times in the recommendation process. Moreover, we labeled the real-time performance by simulators with different colors ("red" for wrong, "black" for right). From the figure, we can see: (1) the difficulty levels of recommendations with DRER do not vary dramatically in most cases, suggesting that the smoothness reward is effective. However, we also notice that this performance in MATH dataset at step 10 is not satisfied. We guess that DRER may not well track the student's state at the beginning so that its recommendations are not stable. (2) If we set g with lower value (0.2), i.e., we hope the student gets more challenging exercising experience, DRER would recommend more difficult exercises so the simulated student would be expected to frequently answer them wrong, showing that the engagement reward works as we expected. With these observations, we can conclude that DRE framework could well adjust recommendations for adapting students with different learning needs in practice.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel *D*eep *R*einforcement learning framework for *E*xercise recommendation, namely DRE. In the framework, we proposed two Exercise Q-Networks (EQN) to select exercise recommendations following different mechanisms, i.e., a straightforward *EQNM with Markov property* and a sophisticated *EQNR with Recurrent manner*. Comparatively, EQNR tracked the dynamic nature of learning process for recommendation, which was superior to EQNM. In addition, we leveraged three domain-specific rewards to characterize the benefits of factors including *Review & Explore*, *Smoothness* and *Engagement*, for letting DRE find the optimal recommendation strategy. Extensive experiments on two real-world datasets demonstrated the effectiveness of DRE.

There are still some interesting directions for further studies. First, We would like to seek more ways to learn the reward settings automatically in practice. For example, we could dynamically adjust the learning goal g in Eq. (9) by monitoring students' behaviors. If the student solves exercises very quickly, then the agent could set g with a lower value to challenge her. Moreover, we are willing to develop a system and apply DRE framework online. Moreover, DRE is a general framework, which has potentials to be extended to many other applications, such as e-commerce, where some kind of users' feedbacks can be equivalent to students' performance. In the future, we are willing to hold our Exercise Q-Networks as well as the three reward objectives for more studies.

ACKNOWLEDGEMENTS

This research was partially supported by grants from the National Key Research and Development Program of China (No. 2018YFC0832101), the National Natural Science Foundation of China (Grants No. U1605251, 61922073), and the Science Foundation of Ministry of Education of China & China Mobile (No. MCM20170507). Zhenya Huang gratefully acknowledges the support of the China Scholarship Council (No. 201806340212).

REFERENCES

[1] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. 2014. Engaging with massive online courses. In *Proceedings of the 23rd international conference on World wide web*. ACM, 687–698.
[2] Richard Bellman. 2013. *Dynamic programming*. Courier Corporation.
[3] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed Chi. 2018. Top-K Off-Policy Correction for a REINFORCE Recommender System. *arXiv preprint arXiv:1812.02353* (2018).
[4] Yunxiao Chen, Xiaoou Li, Jingchen Liu, and et al. 2018. Recommendation System for Adaptive Learning. *Applied psychological measurement* 42, 1 (2018), 24–41.
[5] Yuying Chen, Qi Liu, Zhenya Huang, Le Wu, Enhong Chen, Runze Wu, Yu Su, and Guoping Hu. 2017. Tracking knowledge proficiency of students with educational priors. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 989–998.
[6] Jimmy De La Torre. 2009. DINA model and parameter estimation: A didactic. *Journal of educational and behavioral statistics* 34, 1 (2009), 115–130.
[7] Louis V DiBello, Louis A Roussos, and William Stout. 2006. 31a review of cognitively diagnostic assessment and a summary of psychometric models. *Handbook of statistics* 26 (2006), 979–1030.
[8] Chase Geigle and ChengXiang Zhai. 2017. Modeling MOOC student behavior with two-layer hidden Markov models. In *Proceedings of the fourth (2017) ACM conference on learning@ scale*. ACM, 205–208.
[9] Yoav Goldberg and Omer Levy. 2014. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722* (2014).
[10] Mengyue Hang, Ian Pytlarz, and Jennifer Neville. 2018. Exploring student check-in behavior for improved point-of-interest prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 321–330.
[11] Pedro Hontangas, Vicente Ponsoda, Julio Olea, and Steven L Wise. 2000. The choice of item difficulty in self-adapted testing. *European Journal of Psychological Assessment* 16, 1 (2000), 3.
[12] Zhenya Huang, Qi Liu, Enhong Chen, Hongke Zhao, Mingyong Gao, Si Wei, Yu Su, and Guoping Hu. 2017. Question Difficulty Prediction for READING Problems in Standard Tests.. In *AAAI*. 1352–1359.
[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[14] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.
[15] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 345–354.
[16] Qi Liu, Runze Wu, Enhong Chen, Guandong Xu, Yu Su, Zhigang Chen, and Guoping Hu. 2018. Fuzzy cognitive diagnosis for modelling examinee performance. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 4 (2018), 48.
[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[19] Genevieve B Orr and Klaus-Robert Müller. 2003. *Neural networks: tricks of the trade*. Springer.
[20] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*. 505–513.
[21] q. liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu. 2019. EKT: Exercise-aware Knowledge Tracing for Student Performance Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2019), 1–1. https://doi.org/10.1109/TKDE.2019.2924374
[22] Lawrence Rabiner and B Juang. 1986. An introduction to hidden Markov models. *ieee assp magazine* 3, 1 (1986), 4–16.
[23] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.
[24] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).
[25] Saman Shishehchi, Seyed Yashar Banihashem, Nor Azan Mat Zin, and Shahrul Azman Mohd Noah. 2011. Review of personalized recommendation techniques for learners in e-learning systems. In *2011 international conference on semantic technology and information retrieval*. IEEE, 277–281.
[26] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.
[27] Yu Su, Qingwen Liu, Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Chris Ding, Si Wei, and Guoping Hu. 2018. Exercise-enhanced sequential modeling for student performance prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[28] Steven Tang, Joshua C Peterson, and Zachary A Pardos. 2016. Deep neural networks and how they apply to sequential education data. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. ACM, 321–324.
[29] Xueying Tang, Yunxiao Chen, Xiaoou Li, Jingchen Liu, and Zhiliang Ying. 2018. A reinforcement learning approach to personalized learning recommendation systems. *Brit. J. Math. Statist. Psych.* (2018).
[30] John K Tarus, Zhendong Niu, and Ghulam Mustafa. 2018. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review* 50, 1 (2018), 21–48.
[31] Shan-Yun Teng, Jundong Li, Lo Pang-Yun Ting, Kun-Ta Chuang, and Huan Liu. 2018. Interactive unknowns recommendation in e-learning systems. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 497–506.
[32] Nguyen Thai-Nghe, Lucas Drumond, Artus Krohn-Grimberghe, and Lars Schmidt-Thieme. 2010. Recommender system for predicting student performance. *Procedia Computer Science* 1, 2 (2010), 2811–2819.
[33] Le Wu, Lei Chen, Richang Hong, Yanjie Fu, Xing Xie, and Meng Wang. 2019. A hierarchical attention model for social contextual image recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2019).
[34] Le Wu, Yong Ge, Qi Liu, Enhong Chen, Richang Hong, Junping Du, and Meng Wang. 2017. Modeling the evolution of usersâĂŽ preferences and social links in social networking services. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1240–1253.
[35] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. *arXiv preprint arXiv:1904.10322* (2019).
[36] Chun-Kit Yeung. 2019. Deep-IRT: Make Deep Learning Based Knowledge Tracing Explainable Using Item Response Theory. *arXiv preprint arXiv:1904.11738* (2019).
[37] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *International Conference on Artificial Intelligence in Education*. Springer, 171–180.
[38] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 765–774.
[39] Wayne Xin Zhao, Wenhui Zhang, Yulan He, Xing Xie, and Ji-Rong Wen. 2018. Automatically Learning Topics and Difficulty Levels of Problems in Online Judge Systems. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 27.
[40] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1040–1048.