Adaptive Learning Recommendation Strategy Based on Deep Q-learning

Abstract

Personalized recommendation system adaptive to each learner's own learning paces has been widely adopted in E-learning field. With full utilization of learning behavior data, psychometric assessment models keep track of the learner's proficiency on knowledge points and then the well-designed recommendation strategy selects a sequence of actions to meet the objective of maximizing learner's learning efficiency. This paper proposes a novel adaptive recommendation strategy under the framework of reinforcement learning based on the deep Q-learning algorithms, which are the techniques that contributed to the success of AlphaGo Zero to achieve the super-human level in playing the game of go. The proposed algorithm incorporates an early stopping to account for the possibility that learners may choose to stop learning. It can properly deal with missing data and can handle more individual-specific features for better recommendations. The recommendation strategy guides individual learners with efficient learning paths that vary from person to person. We showcase concrete examples with numeric analysis of substantive learning scenarios to further evaluate the power of the proposed method.

Keywords: adaptive learning, Markov decision process, recommendation system, reinforcement learning.

Adaptive Learning Recommendation Strategy Based on Deep Q-learning

Introduction

Adaptive learning refers to an educational method that delivers personalized educational interventions, including recommendations of learning actions, to learners in order to meet the unique educational needs of the learners. Adaptive learning is typically implemented through computerized algorithms that output personalized action recommendations based on analysis of the learning histories of the learners. The popularization of the Internet access has facilitated the application of adaptive learning in practice, which in turn prompted further research interests (Sleeman and Brown (1982), Wenger (1987)). On the other hand, the past few years have seen great advances in big data technology. In particular, a sensational success was achieved by AlphaGo which employed the cutting-edge techniques on deep reinforcement learning. The aim of this paper is to bridge adaptive learning with the recent developments in deep reinforcement learning and consequently propose novel recommendation strategies for the adaptive learning system.

Emanated from behaviorist psychology (Skinner, 1990), reinforcement learning is concerned about how an agent interacts with the environment and learns to take actions to maximize total rewards. Reinforcement learning is one of the central topics in artificial intelligence (Kaelbling, Littman, & Moore, 1996), and has broad applications to areas such as robotics and industrial automation (Kober & Peters, 2012), health and medicine (Frank, Seeberger, & O'reilly, 2004), and finance (Choi, Laibson, Madrian, & Metrick, 2009), among many others. As a significant progress in recent years, deep reinforcement learning was proposed by Google Deepmind in 2013. In applications to play Atari 2600 games, it achieved the expert human proficiency (Mnih et al., 2013). In applications to playing the game go, it surpassed human level of play, starting from *tabula rasa*, within a training period as short as a few days (Silver et al., 2016).

In order to apply reinforcement learning techniques to adaptive learning, a proper formulation of adaptive learning process in terms of the setup is commonly used in reinforcement learning. As laid out in Chen, Li, Liu, and Ying (2018), an adaptive

learning system consists of three parts, (1) the assessment model that measures the proficiency level of knowledge points through learning behavior data; (2) the learning model that associates the improvement on the corresponding proficiency levels with learning actions; and (3) the recommendation strategy that chooses a sequence of actions to recommend. Following the strategy, the learner takes an action each time, receives a reward and then enters a new knowledge state. Our goal is to help learners master the knowledge points in the most effective path by utilizing all the trackable information up to time t, that is maximizing learning efficiency. Under such optimal strategy, a sequence of actions balances the trade-off between the expected total gain of knowledge points being mastered by the learner and total learning steps during the process. In a substantive learning scenario including assessment errors, unknown learning model and complex reward forms, a good recommendation strategy is supposed to make full use of current information to maximize the learning gain, and is feasible in various learning designs. As in typical reinforcement learning framework, such a strategy design problem optimizing the learning path is modeled as a Markov decision problem.

In this paper, we present an approach based on deep reinforcement learning, giving personalized recommendations and addressing the cost-effective learning need of learners. Specifically, we adopt a variant of deep Q-learning algorithm (Hasselt, Guez, & Silver, 2016), and make tailored modifications for adaptive learning recommendations. The objective function is approximated by a neural network which is used to determine the optimal strategy. Our proposed deep Q algorithm has several novel contributions. First, the parameter space is enlarged and early stopping is incorporated into the learning process. As a result, the deep Q-learning approach maximizes the overall gain within the shortest time, serving the purpose of maximizing the learner's learning efficiency. Second, the model is designed with the ability to handle missing data. In existing work (for example, Chen et al. (2018)), it is assumed that the learning process has fixed procedures, where the assessment model is considered to be indispensable at each step, and the recommendations cannot appropriately deal with missing knowledge status. In this paper, a missing index is introduced and modeled, which helps to

analyze incomplete data in a flexible fashion. Third, the effect of learning interest is considered. Since the learning model differs among learners, a real-time recommendation system that is sensitive to the characteristics of each learner can be further obtained by introducing more personal features, such as the learning interest. Thanks to the nature of the deep neural network, the proposed method can scale up comparatively easily in handling big data. It is expected that, by combining domain knowledge and more individual information, the proposed method may be further developed to a more efficient and competitive adaptive learning system.

The rest of the paper is organized as follows. In Section 2, a mathematical framework for adaptive learning is reviewed, where we define a general cost-efficient reward for our recommendation strategy. Then, a variant of deep Q-learning specially designed for adaptive learning is presented in Section 3. In Section 4, concrete simulations are given to support our methods in the more realistic learning scenarios, followed by the discussion in Section 5.

Background

The objective of recommendation is to help individual learners achieve their learning goals in the shortest time by utilizing all the currently available information. Consider totally K knowledge points with the learning time $t \in [0, T]$. Let $\boldsymbol{s} = (s_1, s_2, \ldots, s_K)$ be the learner's knowledge state, which is a latent dynamic K-dimensional vector. Then $\boldsymbol{s}(t)$ is the knowledge state at time t, of which $s_i(t)$ is the mastery level of the ith knowledge point. Based on the assessment result of a learner's knowledge state $\boldsymbol{s}(t)$ at time t, an appropriate action $a(t)$ is recommended from action space A. In this section, we elaborate the learning procedure in three parts. For a similar framework, see Chen et al. (2018) and Tang, Chen, Li, Liu, and Ying (2018).

Assessment Model

The study on the diagnosis of one's latent abilities has been developed in modern psychometrics (Birnbaum (1968), Reckase (1979), De La Torre and Douglas (2004)). Online learning systems can track the entire online learning behaviors, including

frequency of login, clips on the lecture video, item responses which can be dichotomous or polytomous, etc. Such information can be incorporated to model one's learning styles and preferences so as to make learning designs more accurate and enjoyable for learners (Coffield, Moseley, Hall, & Ecclestone, 2004).

Consider the assessment on the proficiency level of knowledge points with the test item pool τ. The knowledge state $\boldsymbol{s}(t)$ can be partially observed from responses to the test items (i.e., questions). Let Y_j be the response to the jth item with a given mastery for knowledge points $\boldsymbol{s}(t)$ following a distribution, that is

$$Y_j \sim h_{j,\boldsymbol{s}(t)}(y),$$

which implies $h_{j,\boldsymbol{s}(t)}(y)$ depends, in addition to $\boldsymbol{s}(t)$, on a set of item parameters of the jth item. Specifically, $h_{j,\boldsymbol{s}(t)}(y)$ varies based on discrete and continuous latent traits $\boldsymbol{s}(t)$ and different test designs.

We now present a feasible assessment model used in the recommendation design, which considers continuous knowledge states. Assume items are choice questions, designed to have different associated knowledge points. The matrix $\boldsymbol{Q} = (q_{jk})_{|\tau| \times K}$ shows the relationship between the knowledge points and questions (Tatsuoka, 1983). Each row of Q-matrix represents a question, while each column indicates a knowledge point. Specifically, every element of Q-matrix (i.e.,q_{jk}) indicates if one question relates to the knowledge point. For example, $q_{jk} = 0$ means that the jth question cannot measure kth knowledge point at all. Learners are assigned knowledge states based on their item responses and the constructed Q-matrix.

Define $P(Y_j = 1)$ as the probability of one answers jth item correctly, where $Y_j \in \{0, 1\}$. For measuring multiple continuous states, the multidimensional three-parameter logistic (M3PL) IRT model (Birnbaum (1968), Reckase (2009)) is considered, in which the item response distribution takes the form as

$$P(Y_j = 1 | \boldsymbol{s}, \boldsymbol{\alpha_j}, g_j, c_j) = g_j + (1 - g_j) \frac{e^{\boldsymbol{\alpha_j}' \boldsymbol{s} + c_j}}{1 + e^{\boldsymbol{\alpha_j}' \boldsymbol{s} + c_j}}.$$

Here, $g_j \in (0, 1)$ is the guessing parameter, c_j is the intercept parameter and $\boldsymbol{\alpha_j}$ is the K-dimensional discrimination parameter. With learning behavior data, item parameters

can be further updated and revised in return (Lord (1986), Baker and Kim (2004)). Besides the M3PL model, other models including the diagnostic classification model (DCMs; Rupp, Templin, and Henson (2010)), the deterministic input, noisy-And-gate model (DINA; Haertel (1989), Junker and Sijtsma (2001)), the deterministic input and noisy-Or-gate model (DINO; Templin and Henson (2006)) can be adopted as well according to different learning settings. In this paper, we use the M3PL method as the assessment model to estimate the knowledge state and assume the parameters in the assessment model known in all experiments.

Through the learner's item responses, how knowledge points have been mastered so far can be observed partially. The assessment result $\hat{\boldsymbol{s}}$ can be further obtained from either the maximum likelihood estimate or the posterior distribution. In the previous work, the assessment model is assumed to be indispensable in a learning system. However, in practice, it is not a typical case that there always exist interactive learning histories to diagnose $\boldsymbol{s}(t)$ at each t. For example, the learner may skip the questions after learning a material, and the knowledge state $\boldsymbol{s}(t)$ is called the missing data at time t. How to recommend given such incomplete learning data has not been discussed yet. In our paper, the number of missing times is defined as the missing index, which can partially reflects one's learning history. Incorporating the missing index into the design of the recommendation strategy enables the agent to recommend proper actions even though the learner's knowledge state is missing. The detailed method and numerical results are presented in Experiment on Missing Data.

Learning Model

In a learning system, the learning model responses to the action at each time. Given an action $a(t)$ at time t, the knowledge state may have a corresponding change at next time. It is assumed as a Markov process for the transition from $\boldsymbol{s}(t)$ to $\boldsymbol{s}(t+1)$, with probability

$$P(\boldsymbol{s}(t+1) = \tilde{\boldsymbol{s}} | \boldsymbol{s}(t) = \boldsymbol{s}, a(t) = a).$$

In the context of adaptive learning, more factors beyond the learners' knowledge

states should be taken into considerations to learn the learning model comprehensively while learning the recommendation strategy at the same time. These factors including cognitive abilities, learning styles, and other learning behaviors enable to distinguish various learning models for learners. Intuitively, interest in this subject could be a useful feature in the personalized recommendation system. Imagine two learners with different, strong and weak, interest in a course. They are likely to have very different learning processes and outcomes as a result of different interest. In mathematical modeling, different transition probabilities should be assumed for these two learners and different actions may be recommended even if they are at the same knowledge state. We will present later the detailed simulation study to illuminate how interest as a feature affects the learning process and how we can combine such information into recommendation.

Recommendation Strategy

With learners' information up to time t, the recommendation strategy determining actions in the Markov decision process is called policy, denoted as π. We aim to find the optimal policy which maximizes the total gain during the learning process.

By taking the action $a(t)$, the learner will receive an immediate reward $R(t)$, which captures the relatively gain of the action $a(t)$ and is a function of knowledge states. The total gain in a learning trajectory can be quantified by accumulated rewards. Therefore, for $t = 0, 1, \ldots, T$, the optimal strategy π^* is defined as

$$\pi^* = \arg\max_{\pi} E_{\pi}[\sum_{t=0}^{T} R(t)].$$

Given the knowledge state $\boldsymbol{s}(t)$ and action $a(t)$, the quality of this action following the policy π^* can be measured by the Q-value function, i.e.,

$$Q^t_{\pi^*}(\boldsymbol{s}(t), a(t)) = E_{\pi^*}[\sum_{t'=t}^{T} R(t')|\boldsymbol{s}(t) = \boldsymbol{s}, a(t) = a].$$

According to $Q^t_{\pi^*}(\boldsymbol{s}, a)$ at each time, the optimal action is determined from $a^* = \arg\max_a Q^t_{\pi^*}(\boldsymbol{s}, a)$. That is to say, if we know $Q^t_{\pi^*}(\boldsymbol{s}, a)$ for every step, the optimal policy is determined.

Before specifying our strategy, we first talk about some characteristics of a learning system in reality. On the one side, the limited time horizon. The learning time taken on achieving target knowledge state is regarded as an index to evaluate the efficiency of the policy. Thus early stopping should be taken into the recommendation framework. That means a good strategy not only suggests how to learn at least steps, but also recommends the learner to stop if he/she reaches the goal. On the other side, owing to differences in the emphasis on knowledge points, domain knowledge can be incorporated to improve π. For example, the mastery of essential knowledge points should receive more rewards and take more weights than trivial ones when calculating the Q-value function.

In order to find such an efficient learning path to balance total gains on knowledge and learning steps, the action space A includes 3 categories of actions, that is

$$A : \{d_1, d_2, \ldots, d_n, a_s, \text{`null'}\},$$

where d stands for learning materials, a_s suggests stopping learning. After taking a_s, learners will take 'null' action until terminal time. Given the reward at the terminal time (i.e., terminal reward $R(T)$), the reward form is defined as

$$R(t) = \begin{cases} -1, & \text{if taking learning actions } d_1, d_2, \ldots, d_n \text{ at time } t \\ 0, & \text{if taking 'null' or } a_s \\ R(T), & \text{if at the terminal time } t = T. \end{cases}$$

The reward setting imposes the penalty on total rewards by -1 for every step, thus the policy may lead the learning to stop. We consider $R(T) = \phi \sum_{k=1}^{K} w_k s_k(T)$, where ϕ is a scale parameter and w_k is the weight of the kth knowledge point s_k. The idea behind $R(T)$ and ϕ will be further discussed in Learning Scenarios and Experiments. Specifically, the reward can take varied forms, for instance, Chen et al. (2018) and Tang et al. (2018) raised $R(t) = \sum_{k=1}^{K} w_k(s_k(t+1) - s_k(t))$, which intuitively focuses on the difference of knowledge states after learning and visualizes the improvement at each step.

Following the reward setting above, the Q-value function of policy π specifies that

$$Q_{\pi}^{t}(\boldsymbol{s}(t), a(t)) = E_{\pi}[R(T) - \text{learning steps}|\boldsymbol{s}(t) = \boldsymbol{s}, a(t) = a],$$

where 'learning steps' is the number of total steps from time t in a trajectory. A sequence of appropriate actions is chosen by consulting the optimal Q-value function $Q_{\pi^*}^{t}(\boldsymbol{s}, a)$. Therefore, the key problem is transformed from finding π^* to determine the optimal Q-value function at each time.

Consequently, we consider two baselines of recommendation strategies, which serve to illuminate the effect of the proposed strategy. One is the random policy that chooses actions uniformly at random from all available actions. Another one is the oracle policy, where knowledge states can be observed without assessment error and the learning model is known. The oracle policy definitely outperforms other strategies.

Deep Q-learning Recommendation Strategy

Finding the Q-value function with respect to uncertainties in the system is challenging. For traditional methods, dynamic programming (Bellman, 2003) fails due to the curse of dimensionality (Niño-Mora, 2009) inherent in relatively large state spaces. Then approximate dynamic programming methods (Powell, 2007) have constraints on forms of parametrization and may have convergence problems when the form of Q-value function is complicated. In this section, we employ a method based on deep Q-learning (Mnih et al., 2015) to optimize the policy, which is called the DQN method in the rest of the paper.

Q-learning. Q-learning (Watkins & Dayan, 1992) is the basic idea behind the proposed method. In a learning system, a sequence of state transitions $(\hat{\boldsymbol{s}}(t), a(t), R(t), \hat{\boldsymbol{s}}(t+1))$ at each time t is formulated in the Markov decision problem. At each time, the agent selects an action and the learner receives a reward, transits to next knowledge state and then Q-value function is updated. Consider $t = 0, 1, 2, \cdots, T-1$. Starting from a random initialized Q-value function, the Q-value is updated in the ith iteration as follows,

$$Q_{i}^{t}(\hat{\boldsymbol{s}}(t), a(t)) = E[R(t) + \max_{a} Q_{i-1}^{t+1}(\hat{\boldsymbol{s}}(t+1), a)|\boldsymbol{s}(t), a(t)].$$

The equation is derived from the Bellman equation (Sutton & Barto, 1998). We aim to use the above equation as an iterative update to obtain the optimal Q-values for all possible actions at each time.

Deep Q-network. For large state space, it is often impractical to maintain the Q-values for all state-action pairs. We consider to approximate $Q^t(\hat{\boldsymbol{s}}(t), a)$ using a parameterized nonlinear function $Q(\hat{\boldsymbol{s}}(t), a, t; \boldsymbol{\theta})$. According to the current time t and state estimator $\hat{\boldsymbol{s}}(t)$, the DQN method predicts $Q^t(\hat{\boldsymbol{s}}(t), a)$ by a feed-forward neural network with weight parameter $\boldsymbol{\theta}$, which outputs the Q-values for all possible actions $a \in A$ simultaneously. As universal approximators, the networks organized by layers have a hierarchical structure which makes them well adapted to learn the hierarchies of information, especially such dynamic problems. What's more, the architecture of the networks is flexible, which can be scaled up and aligned to specific problems. For example, the structure of the network in our experiments is shown in Figure 1, which has two hidden layers and the rectified linear unit (ReLU) serving as the activation function, that is $ReLU(x) = \max(0, x)$. It is worthy to point out that the action is not incorporated as the input of the network and the network outputs all Q-values for current state at each time.

Algorithm. We elaborate the deep Q-learning algorithm in this part. The learner's learning experiences at each time, $\boldsymbol{e}_t = (\hat{\boldsymbol{s}}(t), a(t), R(t), \hat{\boldsymbol{s}}(t+1))$, are stored in the memory D with size N, $D = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_N\}$. We apply mini-batch updates, a batch of samples, $\boldsymbol{e} \sim D$, drawn at random from D and used to optimize the parametrization. In the iteration i, we update $\boldsymbol{\theta}$ by reducing the error between the predicted value $Q_i(\hat{\boldsymbol{s}}(t), a, t; \boldsymbol{\theta}_i)$ of current state and the target Q-value given by the sum of $R(t)$ and the maximized Q-value of the next state $\max_a Q_{i-1}(\hat{\boldsymbol{s}}(t+1), a, t+1; \boldsymbol{\theta}_i^-)$, where $\boldsymbol{\theta}_i^-$ is from the previous iteration. Thus, the loss function is defined as

$$L(\boldsymbol{\theta}_i) = E_{\hat{\boldsymbol{s}}, a}[(y_i - Q_i(\hat{\boldsymbol{s}}(t), a(t), t; \boldsymbol{\theta}_i))^2],$$

where $y_i = R(t) + \max_a Q_{i-1}(\hat{\boldsymbol{s}}(t+1), a, t+1; \boldsymbol{\theta}_i^-)$. Note that the parameter $\boldsymbol{\theta}_i^-$ from previous iteration is held fixed when optimizing $L(\boldsymbol{\theta}_i)$ and updated after constant C steps.

The training procedure is briefly summarized as follows.

(1). Initialize the Q-value function Q with parameter $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{-} = \boldsymbol{\theta}$;

(2). Take action $a(t)$ according to the ϵ-greedy policy;

(3). Store transition $(\hat{\boldsymbol{s}}(t), a(t), R(t), \hat{\boldsymbol{s}}(t+1))$ of time t in the memory D;

(4). Sample a mini-batch of transitions from D;

(5). Compute the predicted Q-value with $\boldsymbol{\theta}$ and the target Q-value with $\boldsymbol{\theta}^{-}$;

(6). Update parameter $\boldsymbol{\theta}$ by optimizing the loss function given the target and prediction;

(7). Every C steps, reset $\boldsymbol{\theta}^{-} = \boldsymbol{\theta}$;

(8). Repeat (2)-(7).

The complete algorithm for training the deep Q-network and more training details are presented in Appendix.

Remark. We provide some intuitions and ideas about the algorithm below.

• Each transition of experiences is potentially used in many parameter updates, which allows for greater data efficiency. Besides, drawing transitions randomly from D not only breaks correlations between the samples but also smooths out the learning and avoids oscillations during training (Mnih et al., 2013).

• The dynamic ϵ-greedy policy (Sutton & Barto, 1998) in (2) allows for an adequate exploration of the state space. The agent takes the optimal action with probability $1 - \epsilon$ and selects a random action with probability ϵ. Starting from a large initialized value, the exploration rate ϵ will gradually decrease as the number of episodes increases, and finally guarantees full utilization of high-payoff actions.

• Periodically updating the parameter when computing the target Q-value in (7) avoids the constantly shift of the target Q-value and further mitigates the risk that the network falls into feedback loops between the target and estimated Q-values in the training (Mnih et al., 2015).

Reinforcement learning refers to goal-oriented algorithms and Figure 2 visualizes that how it works in a flow chart. The DQN method starts from an initialized Q-value function and improves the policy design according to the interactions with the

environment. By means of DQN, the system can handle a relatively large state space and extract essential information from learning behavior data.

Learning Scenarios and Experiments

In this section, we present a concrete learning system, conduct simulations to validate the efficiency of our recommendation strategy in different learning scenarios and explore the effect of learning interest by combining the classification problem in the DQN method. Review that a special terminal reward takes the form as

$$R(T) = \phi \boldsymbol{w}' \boldsymbol{s}(T),$$

where ϕ is the scale parameter and $\boldsymbol{w}'\boldsymbol{s}(T)$ is the weighted achievement on all knowledge points, i.e., $\sum_{k=1}^{K} w_k s_k(T)$, w_k is the weight of the kth knowledge point s_k. Therefore we rewrite the reward as

$$R(t) = \begin{cases} -1, & \text{if taking learning actions } d_1, d_2, \ldots, d_n \text{ at time } t \\ 0, & \text{if taking 'null' or } a_s \\ \phi \boldsymbol{w}' \boldsymbol{s}(T), & \text{if at the terminal time } t = T. \end{cases}$$

The terminal reward can be defined in other ways to meet varied learning goals. We provide some intuitions about our reward setting below.

(1). Suppose there indeed exists a final exam at the terminal time and the final grade is a value measuring learners' proficiency. If the K-dimensional weight $\boldsymbol{w}$ represents the importance of knowledge points, then final grade can be expressed by $\boldsymbol{w}'\boldsymbol{s}(T)$, which refers to the weighted achievement after the whole learning process.

(2). Let ϕ to be a scale parameter of the final grade, regarded as a character index, showing the degree of the learner's willing to achieve better. Imagine a course is very important and the learner views the final grade heavily. It is worth the efforts and thus a large ϕ makes sure that the learner will take enough learning steps to master almost all the knowledge points for a good final grade. In this case, the terminal reward $R(T)$ can compensate the cost from learning steps. The scale can be determined by the learner's self-evaluation, and even set fixed by the instructor in some learning scenarios.

Toy Experiment

We firstly go through a toy example, where the oracle policy is known. Assume a course consisting of three knowledge points ($K = 3$), corresponding to two proficiency levels in the Markov learning model, i.e., $\boldsymbol{s} \in \{0, 1\}^3$ where 0 means not mastered yet and otherwise, 1. Besides, the knowledge points are hierarchical. Specifically, mastering point 1, that is $s_1 = 1$, is the prerequisite to learn point 2 and then point 3. As shown in Figure 3, the hierarchy can be reflected in a chain structure. Based on the relationships among the knowledge points, there are only four possible knowledge states $(0, 0, 0), (1, 0, 0), (1, 1, 0), (1, 1, 1)$, denoted as $\boldsymbol{S}_1, \boldsymbol{S}_2, \boldsymbol{S}_3$ and $\boldsymbol{S}_4$ respectively.

Consider three lecture materials $\{d_1, d_2, d_3\}$ relate to three points. Then the transition matrices given learning actions, d_1, d_2, d_3, can be represented as

$$\boldsymbol{P}^{\boldsymbol{d_1}} = \begin{bmatrix} 0.3 & 0.7 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, \boldsymbol{P}^{\boldsymbol{d_2}} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.5 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, \boldsymbol{P}^{\boldsymbol{d_3}} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.7 & 0.3 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix},$$

which are the 4×4 matrices corresponding to 4 possible knowledge states. The transition matrices indicate transition probabilities by taking actions. For example, if a learner is at $\boldsymbol{S}_1 = (0, 0, 0)$ and takes the action d_1, he/she has 0.7 chance to master knowledge 1 and transits to next state $\boldsymbol{S}_2 = (1, 0, 0)$, that is $\boldsymbol{P}^{\boldsymbol{d_1}}{}_{12} = P(\boldsymbol{s}(t+1) = \boldsymbol{S}_2 | \boldsymbol{s}(t) = \boldsymbol{S}_1, a(t) = d_1) = 0.7$. In addition, there exists an underlying assumption in the transition matrices that once a knowledge point is mastered, there is no retrograde anymore.

The weight of knowledge points $\boldsymbol{w}$ is set to be $(0.6, 0.25, 0.15)'$, showing knowledge point 1 takes 60% weight in the final exam. We set terminal time $T = 10$ to ensure we can observe the early stopping phenomena.

We conduct 6 strategies in total. Notice that the assessment model is simulated to obtain estimator $\hat{s}$ at each step. The items measure the knowledge points that relate to the corresponding learning action. For example, if one student takes the learning action

d_1 which can only improve s_1, then the following items are all associated with s_1. The more items (i.e. questions) students do, the assessment model can get more accurate estimators of current states. For the strategies with the IRT model, we conduct three experiments denoted as IRT_2, IRT_8, and IRT_64 where a total of 2, 8 and 64 items at each step are used to estimate knowledge states. Since we focus on the recommendation strategy but not the assessment, an additional experiment where knowledge states can be observed without assessment error is conducted as well, denoted as NOIRT. The policies in these four experiments are designed by the DQN method and we include the random and oracle policy as baselines.

Simulation settings. In the simulation, we apply the DQN method to make the recommendation, where the feed-forward neural network with 2 hidden layers is used to approximate the Q-value function. The Adam technique (Kingma & Ba, 2014) is adopted to optimize $\boldsymbol{\theta}$ with batch size 64. Besides that, we set memory capacity $|D| = 30000$, the initial exploration rate $\epsilon_0 = 0.9$, the end rate $\epsilon_M = 0.05$ and decay of rate $\tau = 3000$. Given the initial knowledge state $\boldsymbol{s}(0) = (0, 0, 0)$, we conduct 15,000 episodes in the training and test the policy over 300 trajectories.

Criteria. To show the power of our DQN method, the random policy serves as a lower benchmark while the oracle policy is the upper benchmark. The performances of the recommendation strategies are evaluated by total rewards received in the learning processes. Given a scale parameter ϕ, the higher total rewards are obtained, the better the strategy is.

Simulation results. Figure 4 shows the results of the toy example. As we can see in the left one, total rewards at different scales in 6 simulations are given. The NOIRT curve with our DQN strategy and the oracle curve almost coincide. The other DQN methods with IRT can still work well with relatively high rewards in the end. Moreover, we have an intuitive conclusion that the model including more items can lead to a better result from comparisons among IRT_2, IRT_8, and IRT_64. For clarity of expression, we rescale the final grade $\boldsymbol{w}'\boldsymbol{s}(T)$ to 100 in the middle figure. It shows that the final grade increases along with the scale parameter ϕ increases. When the scale is

set to be around 23, the final grade in the oracle, NOIRT and IRT_64 can all rise up to 90. The right figure presents the learning steps at different scales in these 6 experiments, where the scale ϕ explores how early stopping occurs in the learning process. When the scale is relatively small, the learner is suggested to stop in the early stages. Due to the randomness and imprecise assessments in the simulation, the IRT_2 curve takes more steps than expectation and fails to stop early.

In summary, given reliable state estimators, our recommendation strategies with the DQN method work well. Moreover, based on differences of learners' willings to learn, our approach not only leads to personalized learning paths but also recommends the stop action at a proper time, which improves the efficiency of learning and increases utilization of time.

Continuous Case

We next provide a more practical learning environment, where the continuous state space is considered. In this case, the proficiency level of each knowledge point is no longer roughly divided into 0 or 1, but an exact continuous value in the interval $[0, 1]$.

The learning system consists of 10 knowledge points ($K = 10$) marked by $1, 2, \ldots, 10$ and the knowledge graph in Figure 5 describes the hierarchical relationships and constraints on learning certain points. The number on each arrow indicates the prerequisite of the proficiency for certain knowledge points. For instance, the number 0.2 on the arrow from 1 to 4 means that the point 4 is assumed to be learned unless the knowledge state of point 1 is larger than 0.2. These constraints imposed on the knowledge structure make the learning model close to reality and more complicated. For ease of training, we summarize the prerequisites into a function $P(\cdot)$ and include $P(\cdot)$ in the environment. $P(\cdot)$ maps a knowledge state to a 10-dimensional zero-one vector where one means the learner is qualified to learn that corresponding knowledge point. For example, input the state $\boldsymbol{s} = (0.4, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and a vector is obtained through $P(\boldsymbol{s}) = (1, 0, 0, 1, 0, 0, 0, 0, 0, 0)$, indicating the proficiency of knowledge point 1 is 0.4 and the learner is able to learn point 4.

For learning actions, a total of 30 materials, i.e., $\{d_1, d_2, \ldots, d_{30}\}$ are generated, of which three materials are associated with one related knowledge point, seven materials with two points, ten materials with three points and the rest of materials with all points. As designed in the toy example, items given in the assessment at each step measure the knowledge points related to the corresponding learning action. We denote $\boldsymbol{W}_a$ as the K-dimensional weight of all knowledge points for the learning action $a \in \{d_1, d_2, \ldots, d_{30}\}$.

We take the transition kernel as a density function to reduce the storage space. Assume the learner takes action $a(t)$ at time t and the transition function is defined as follows,

$$\boldsymbol{s}(t+1) = 1 - (1 - \boldsymbol{s}(t)) \odot \exp\{-2\xi \cdot \boldsymbol{W}_{a(t)} \odot P(\boldsymbol{s}(t))\}, \tag{1}$$

where $\odot$ stands for element-wise multiplication of vectors and $\xi \sim \chi^2_m$, m is the degree of freedom meaning the extent of learning efficiency. The randomness in the learning model is reflected in ξ and a large m stands for a quick and effective learner. We set $m = 2$ here and in the last experiment, we use different m in the transition function to represent two types of learners and combine the classification problem with the DQN method.

The transition function shows some properties of the learning model. On the one hand, the exponential term is always smaller than one, which implies no retrograde in the learning. On the other hand, we can rewrite the equation (1) as,

$$\frac{1 - \boldsymbol{s}(t+1)}{1 - \boldsymbol{s}(t)} = \exp\{-2\xi \cdot \boldsymbol{W}_{a(t)} \odot P(\boldsymbol{s}(t))\}.$$

The above equation shows that at the initial stage, learners always acquire knowledge quickly and knowledge states have a fast increase. On the contrary, it is hard to have a big improvement when the state closes to 1. This phenomenon is a common occurrence in the learning process. It is easy to have a basic understanding for beginners while doing some in-depth work on that basis is rather hard and effortful. The hyperparameters in the simulation setting keep same in the toy example. Finally, we set

the initial knowledge state $\boldsymbol{s}(0)$ to be $(0,0,0,0,0,0,0,0,0,0)'$,

$$\boldsymbol{w} = (0.05, 0.1, 0.05, 0.1, 0.1, 0.2, 0.15, 0.1, 0.1, 0.05)'$$

as the terminal weight of knowledge points and terminal time $T = 20$.

Criteria. 5 simulations under above settings are conducted to compare with the random policy when the knowledge state space is continuous. The network architecture remains unchanged. Due to the complicated learning model, it is hard to dig out the oracle policy at every scale ϕ and we do not present oracle policy in this part. Instead, we compare the DQN method with the linear approximation, where the Q-value function is approximated by a weighted linear sum of knowledge states (Melo & Ribeiro, 2007). In the linear approximation method, 8 items in the IRT model are used to estimate the state at each step. The same three indicators including expected total rewards, the final grade and learning steps will be given to evaluate the efficiency of strategies.

Simulation results. The results are shown in Figure 6, where we generate 30,000 episodes in the training and test on over 1,000 trajectories. Besides strategies mentioned in the toy example, the new curve named Linear presents results from the linear approximation. The results are similar with previous results in the toy example. Without assessment error, the NOIRT curve has the best performance over others and receives the highest final grade but the least learning steps. The left graph presents an intuitive performance, showing the more accurate measurement lays a foundation for a better recommendation. In terms of the linear approximation method, although it can beat random policy, compared with DQN method, the linear approximation cannot fit Q-value function well when the form of Q-value function is complicated and thus does not have a solid performance.

In summary, the DQN method outperforms the random policy and linear approximation method in the above setting, even in the case of IRT_2. Considering such a large state space, our method has advantages in learning from experiences and is not as limited to the parametrization forms as the linear approximation method.

Experiment on Missing Data

Generally, the diagnosis and recommendation for a learner in an adaptive learning system have fixed procedures as shown in Figure 7. The relationships among these three models support such a dynamic learning environment: (1) The knowledge state $\boldsymbol{s}(t)$ is partially observed by responses, estimated as $\hat{\boldsymbol{s}}(t)$ in the assessment model; (2) Given $\hat{\boldsymbol{s}}(t)$, action $a(t)$ is recommended following the policy π; and (3) According to the learning action $a(t)$, the learning model determines the next knowledge state $\boldsymbol{s}(t+1)$. However, students are likely to skip exercises, and thus we cannot observe their knowledge states through the assessment model. Forsaking above fixed procedures, the case when $\hat{\boldsymbol{s}}(t)$ is missing should be dealt and an appropriate action will still be recommended. In this part, we provide a simulation study on incomplete learning data. Here we consider the continuous state space and use the same background setting as the previous experiment.

With the full utilization of current information, we take the number of missing times as a missing index. When the learner skips the questions at time t after taking a learning action, the missing index at t counts 1. If the learner skips the assessment part again at $t+1$, the missing index is added up to 2 and so on. Once the knowledge state can be observed, the missing index will be reset to be 0. Note that the missing index can partially reflect the efforts the learner has paid during the unobserved state period. Given this feature, we expand the input dimension by adding a missing index. The input of the neural network is from $(K+1)$-dimensional to $(K+2)$-dimensional and the input is 12-dimensional in this simulation. The architecture of the neural network is the same as previous simulations.

Criteria. We compare the expected total rewards of DQN strategies based on varying percents of the missing data against the random policy. Particularly, we take 0%, 20%, 40%, 60%, 80% of knowledge states as missing data respectively, to show the performance of the DQN method through experiments. The assessment model with 16 items is given in all recommendation strategies.

Simulation results. We present the results in Figure 8. The left figure shows that compared with the 0% curve (i.e., $\hat{\boldsymbol{s}}$ is known at each step), other missing cases can achieve almost same total rewards at T. With increasing missing levels, more learning steps are taken and lower total rewards are obtained. It reveals that the DQN method extracts efficient learning information from the missing index, although the knowledge state is unobserved. In these cases, the DQN method can properly deal with the missing data since it learns from enough learning experience and recommends the learner to make more efforts for targets.

It is necessary to point out several exceptions, for example when scale parameter $\phi = 26$, the expected total rewards of 40% missing data are higher than 20%. It occurs because of randomness in the simulation. When applying in practice, we need enough interactions with environment and train several independent neural networks to get a stable result.

Experiment on the Effect of Learning Interest

The learning model differs among learners and in order to make a good recommendation, more features added to distinguish the type of learners can further improve the policy design. Such features in a broad sense including gender, mother languages even past grades contain individual-specific learning information. Intuitively, the interest in the subject can be an obvious personal feature for a learner. Under this consideration, we combine the reinforcement learning and classification problem by adding the interest into the approximation of Q-value function to learn the learning model comprehensively.

We simulate two types of learners, type 1 and type 2 and they have different transition probabilities for different learning efficiency. For learners of type 1 with low learning efficiency, ξ in (1) follows χ^2_1, while χ^2_8 for efficient learners of type 2. Notice that the learning model is unknown in the implementation. The learners' interest in the subject can be easily tracked by a questionnaire in the learning system. In the simulation, we consider two choices are given in the questionnaire: 'interested',

'uninterested'. Learners interested in the subject are assumed to be from type 1 with probability 0.1, while uninterested learners are from type 1 with probability 0.9. That means the learning interest reflects the learning efficiency of learners and then we introduce the effect of learning interest to classify the learning model better and assist the decision-making simultaneously.

Similar to the missing data experiment, we consider the interest as an additional dimension in the input which is 12-dimensional. The architecture of the neural network is the same as previous simulations. Although more specific information is included into training, the approximation of the optimal strategy is computationally feasible and easy to be implemented by the DQN method.

Criteria. Three recommendation strategies are conducted. In order to provide a control case to validate whether adding a personal feature can improve the recommendation strategy, we apply the DQN method in the model involving the interest feature, denoted as interest and also in the non-interest model. The neural networks have the same architecture but different dimensions of input. Besides, the experiment where types of learners are already known is also presented as the upper benchmark, denoted as type_classified. To validate the interest effect and have a clear comparison, we consider the knowledge states are measured without error in these three simulations.

Simulation results. The results are presented in Figure 9. From the results of expected total rewards, our model involving interest effect is better than the control case. It demonstrates that the DQN method learns from the interest feature and successfully distinguishes these two types of students to some extent. In this experiment, the flexibility of network helps the agent to further explore the learning model in a simple manner. In other words, it avoids adding high-level terms in the parametrization and solving a considerably high-dimensional scheduling problem in traditional methods.

We elaborate the effect of learning interest as an example to show the efficiency and feasibility of the DQN method in learning features. Of course, one feature is not enough to learn the learning model comprehensively and we should consider more personal features to make a precise recommendation.

Discussion

In this paper, we consider the DQN recommendation strategy by making full use of available information under the framework of the adaptive learning system. We introduce three components of learning system and formulate our problem which is to optimize a Markov decision problem. Due to the cost-efficient reward we set and the complex structure of Q-value function with relatively large parameter space, a feed-forward neural network is used to approximate the objective function and we train the optimal policy by the deep Q-learning method. The current work focuses on developing a general method to make recommendations in the substantive learning scenarios. Early stopping is discussed and we also handle missing data when there exist unfixed learning procedures. Besides, the effect of learning interest is incorporated into Q-value function approximation as well, leading to a more personalized strategy. We showcase four concrete examples in the simulation to validate the power of the DQN method.

In the future work, more practical issues need to be considered. As illuminated in the first two simulations, the accuracy of knowledge state estimators can affect the quality of the recommendation strategy to some extent. Even though the assessment model is not paid much attention here, more individual-specific information can further improve the estimation accuracy. For example, cognitive skills can also be modeled by psychological tests (Brown & Burton, 1978). Combining learning materials designs, the recommendation can not only help to master knowledge but also strengthen such cognitive abilities. On the other side, how to define the reward can be improved in the meantime. Except the reward setting in this paper, the terminal reward can be defined in other forms to measure the feedback of a sequence of actions, for instance, it may directly relates to the learning duration.

Come back to deep learning methods, classical methods such as dynamic programming, are put away for their constraints in the adaptive learning. The deep Q-learning method in practice needs to collect data and train the optimal strategy simultaneously in the initial stage, where the initial policy may be uncomfortable for

learners. Some prior information to build the initial policy can improve the user experience. What's more, a virtual but reasonable learning model serving as the environment can simulate training data and do a major. As a data-driven approach, it is of interest to explore more effective networks to extract behavior information. For example, the number of layers in the neural network can scale up to meet more complicated situations. Besides, the method of policy gradient (Sutton, McAllester, Singh, & Mansour, 1999) may be borrowed, by which a more flexible stochastic policy can be obtained. Finally, theoretical interpretations remain to be studied to prove the feasibility of the deep Q-learning method in the practical large-scale learning environment.

References

Baker, F. B., & Kim, S.-H. (2004). *Item response theory: Parameter estimation techniques*. Boca Raton, FL: CRC Press.

Bellman, R. E. (2003). *Dynamic programming*. New York, NY: Dover Publications, Inc.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. *Statistical theories of mental test scores*, 397–479.

Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive science*, *2*(2), 155–192.

Chen, Y., Li, X., Liu, J., & Ying, Z. (2018). Recommendation system for adaptive learning. *Applied psychological measurement*, *42*(1), 24–41.

Choi, J. J., Laibson, D., Madrian, B. C., & Metrick, A. (2009). Reinforcement learning and savings behavior. *The Journal of finance*, *64*(6), 2515–2534.

Coffield, F., Moseley, D., Hall, E., & Ecclestone, K. (2004). *Learning styles and pedagogy in post-16 learning: a systematic and critical review*. London: Learning and Skills Research Centre.

De La Torre, J., & Douglas, J. A. (2004). Higher-order latent trait models for cognitive diagnosis. *Psychometrika*, *69*(3), 333–353.

Frank, M. J., Seeberger, L. C., & O'reilly, R. C. (2004). By carrot or by stick: cognitive reinforcement learning in parkinsonism. *Science*, *306*(5703), 1940–1943.

Haertel, E. H. (1989). Using restricted latent class models to map the skill structure of achievement items. *Journal of Educational Measurement*, *26*(4), 301–321.

Hasselt, H. v., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the thirtieth aaai conference on artificial intelligence* (pp. 2094–2100). AAAI Press. Retrieved from `http://dl.acm.org/citation.cfm?id=3016100.3016191`

Junker, B. W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, *25*(3), 258–272.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A

survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kober, J., & Peters, J. (2012). Reinforcement learning in robotics: A survey. In M. Wiering & M. van Otterlo (Eds.), *Reinforcement learning: State-of-the-art* (pp. 579–610). Berlin, Heidelberg: Springer Berlin Heidelberg.

Lord, F. M. (1986). Maximum likelihood and bayesian parameter estimation in item response theory. *Journal of Educational Measurement*, *23*(2), 157–162.

Melo, F. S., & Ribeiro, M. I. (2007). Q-learning with linear function approximation. In N. H. Bshouty & C. Gentile (Eds.), *Learning theory* (pp. 308–322). Berlin, Heidelberg: Springer Berlin Heidelberg.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Niño-Mora, J. (2009). A restless bandit marginal productivity index for opportunistic spectrum access with sensing errors. In R. Núñez-Queija & J. Resing (Eds.), *Network control and optimization* (pp. 60–74). Berlin, Heidelberg: Springer Berlin Heidelberg.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality (wiley series in probability and statistics)*. New York, NY: Wiley-Interscience.

Reckase, M. D. (1979). Unifactor latent trait models applied to multifactor tests: Results and implications. *Journal of educational statistics*, *4*(3), 207–230.

Reckase, M. D. (2009). *Multidimensional item response theory* (Vol. 150). New York, NY: Springer.

Rupp, A. A., Templin, J., & Henson, R. A. (2010). *Diagnostic measurement: Theory,*

methods, and applications. New York, NY: Guilford Press.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., van den Driessche, G., ... Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Skinner, B. F. (1990). *The behavior of organisms: An experimental analysis*. Cambridge, MA: BF Skinner Foundation.

Sleeman, D., & Brown, J. S. (1982). *Intelligent Tutoring Systems*. London: Academic Press. Retrieved from `https://hal.archives-ouvertes.fr/hal-00702997`

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1) (No. 1). Cambridge, MA: MIT press Cambridge.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th international conference on neural information processing systems* (pp. 1057–1063). Cambridge, MA: MIT Press. Retrieved from `http://dl.acm.org/citation.cfm?id=3009657.3009806`

Tang, X., Chen, Y., Li, X., Liu, J., & Ying, Z. (2018). A reinforcement learning approach to personalized learning recommendation systems. *British Journal of Mathematical and Statistical Psychology*, *0*(0).

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of educational measurement*, *20*(4), 345–354.

Templin, J. L., & Henson, R. A. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological methods*, *11*(3), 287–305.

Thrun, S., & Schwartz, A. (1993, January). Issues in using function approximation for reinforcement learning. In D. T. J. E. M. Mozer P. Smolensky & A. Weigend (Eds.), *Proceedings of the 1993 connectionist models summer school*. Mahwah, NJ: Erlbaum Associates.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge*. San Francisco, CA:

Morgan Kaufmann Publishers Inc.

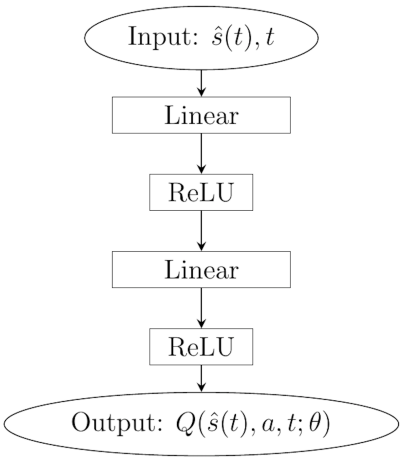

Figure 1. The architecture of the neural network in our experiments.

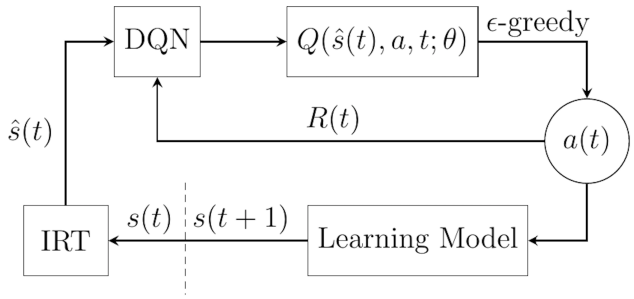

Figure 2. The interactions between the agent and environment with the DQN method.

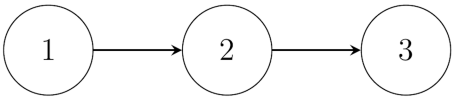

Figure 3. The hierarchy of knowledge points in the toy example.

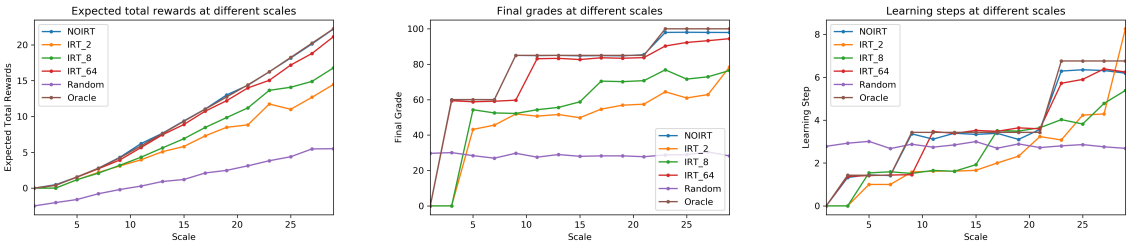

Figure 4. The results of the experiments in the toy example.

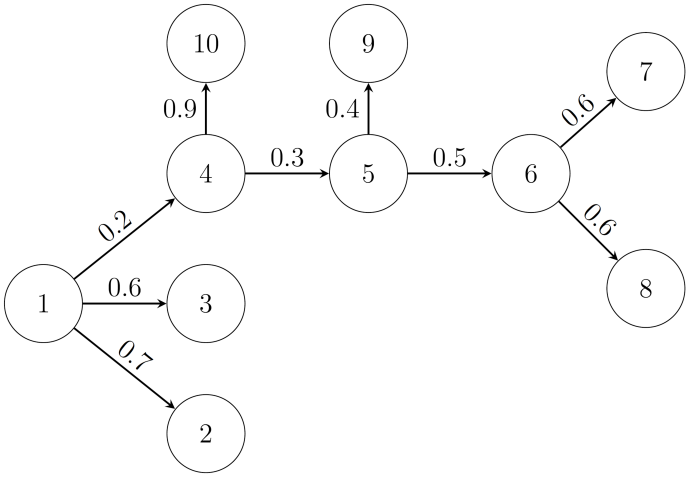

Figure 5. The knowledge graph in the continuous case.

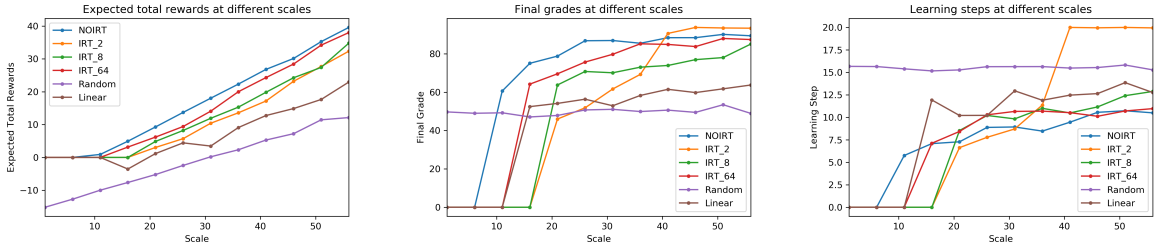

Figure 6. The results of the experiments in the continuous case.

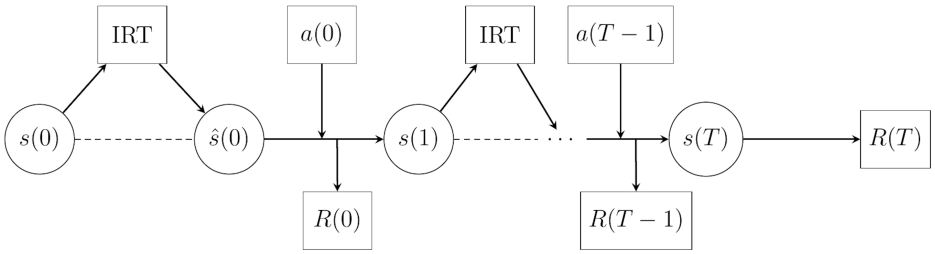

Figure 7. The flow chart of general procedures in the adaptive learning system.

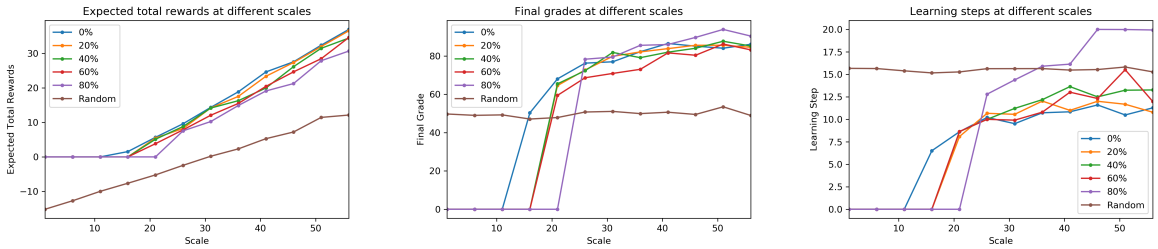

Figure 8. The results of the experiments on the missing data.

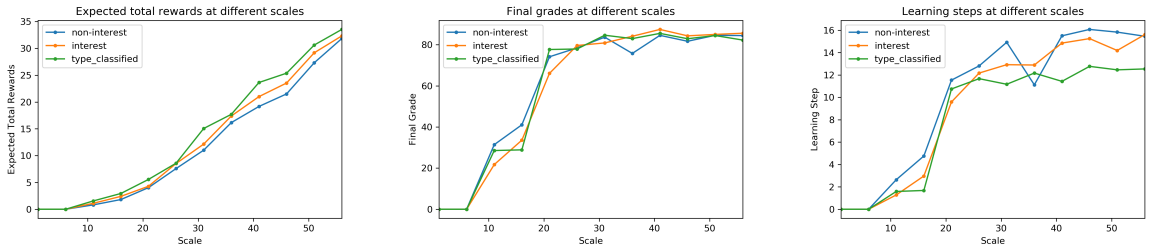

Figure 9. The results of the experiments on the effect of learning interest.

Appendix

Remark. We elaborate some training details below.

• $\boldsymbol{\theta}$ is updated for each iteration following the gradient of $L(\boldsymbol{\theta})$, using the method of stochastic gradient descent with Adam (Kingma & Ba, 2014).

• In order to mitigate the overestimate issue in predicting Q-values (Thrun & Schwartz, 1993) for the DQN method, we use the approach proposed by Hasselt et al. (2016) and rewrite the target as

$$y_i = R(t) + Q_{i-1}(\hat{\boldsymbol{s}}(t+1), \arg\max_a Q_i(\hat{\boldsymbol{s}}(t+1), a, t+1; \boldsymbol{\theta}_i), t+1; \boldsymbol{\theta}_i^-).$$

It decomposes the max operation in the target into the action selection, resulting in more stable and reliable results.

• Since the policy is stochastic, we conduct several independent networks to reduce effects from randomness.

Algorithm 1 Double DQN with assessment model

Input: Begin exploration rate ϵ_0; Decay of rate τ; End learning rate ϵ_M; Learning rate η; Initialized knowledge state $\boldsymbol{s}(0)$; Batch size n;

Output: Action-Value function Q;

Initialize replay memory capacity D to capacity N;

Initialize action-value function Q with random weights $\boldsymbol{\theta}$ and target action-value function $\hat{Q}$ with weights $\boldsymbol{\theta}^{-} = \boldsymbol{\theta}$;

for episode = 1,. . . , M **do**

  Initialize knowledge state $\hat{\boldsymbol{s}}(0) = \boldsymbol{s}(0)$;

  $\epsilon = \epsilon_M + (\epsilon_0 - \epsilon_M) * \exp\{-\text{episode}/\tau\}$;

  for $t = 0, \ldots, T-1$ **do**

    Set $r(t) = \begin{cases} R(T) + R(T-1), & \text{if } t = T-1 \\ R(t), & \text{otherwise} \end{cases}$

    if have not already taking action a_s **then**

      With probability ϵ select a random action $a(t)$;

      otherwise select $a(t) = \arg\max_a Q(\hat{\boldsymbol{s}}(t), a, t; \boldsymbol{\theta})$;

    else

      $a(t) =$ 'null'

    Execute action $a(t)$ in emulator, observe reward $r(t)$ and estimate the next knowledge state $\hat{\boldsymbol{s}}(t+1)$ through assessment model;

    Store transition $(\hat{\boldsymbol{s}}(t), a(t), r(t), \hat{\boldsymbol{s}}(t+1))$ in D;

    Sample random minibatch with size n of transitions $(\hat{\boldsymbol{s}}(t), a(t), r(t), \hat{\boldsymbol{s}}(t+1))$ from D;

    Select $a' = \arg\max_a Q(\hat{\boldsymbol{s}}(t+1), a, t+1; \boldsymbol{\theta})$;

    Set $y(t) = \begin{cases} r(t), & \text{if } t = T-1 \\ r(t) + \hat{Q}(\hat{\boldsymbol{s}}(t+1), a', t+1; \boldsymbol{\theta}^{-}), & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $(y(t) - Q(\hat{\boldsymbol{s}}(t), a(t), t; \boldsymbol{\theta}))^2$ with respect to the network parameters $\boldsymbol{\theta}$ with learning rate η;

    Every C steps reset $\hat{Q} = Q$;

  end for

end for

return $Q(\hat{\boldsymbol{s}}(t), a, t; \boldsymbol{\theta})$;