Article

An Actor-Critic Hierarchical Reinforcement Learning Model for Course Recommendation

Kun Liang, Guoqiang Zhang *, Jinhui Guo and Wentao Li

College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin 300457, China; liangkun@tust.edu.cn (K.L.); guowudi@mail.tust.edu.cn (J.G.); liwentao@mail.tust.edu.cn (W.L.)
* Correspondence: zhangguoqiang@mail.tust.edu.cn

Abstract: Online learning platforms provide diverse course resources, but this often results in the issue of information overload. Learners always want to learn courses that are appropriate for their knowledge level and preferences quickly and accurately. Effective course recommendation plays a key role in helping learners select appropriate courses and improving the efficiency of online learning. However, when a user is enrolled in multiple courses, existing course recommendation methods face the challenge of accurately recommending the target course that is most relevant to the user because of the noise courses. In this paper, we propose a novel reinforcement learning model named Actor-Critic Hierarchical Reinforcement Learning (ACHRL). The model incorporates the actor-critic method to construct the profile reviser. This can remove noise courses and make personalized course recommendations effectively. Furthermore, we propose a policy gradient based on the temporal difference error to reduce the variance in the training process, to speed up the convergence of the model, and to improve the accuracy of the recommendation. We evaluate the proposed model using two real datasets, and the experimental results show that the proposed model significantly outperforms the existing recommendation models (improving 3.77% to 13.66% in terms of HR@5).

Keywords: course recommendation; actor-critic method; hierarchical reinforcement learning method; policy gradient method

check for updates

Citation: Liang, K.; Zhang, G.; Guo, J.; Li, W. An Actor-Critic Hierarchical Reinforcement Learning Model for Course Recommendation. *Electronics* **2023**, *12*, 4939. https://doi.org/10.3390/electronics12244939

Academic Editors: Jesús G. Boticario, Jose M. Alcaraz Calero and David Martín Gómez

Received: 19 October 2023
Revised: 1 December 2023
Accepted: 6 December 2023
Published: 8 December 2023

1. Introduction

Online course education has been developing rapidly, and massive open online courses (MOOCs) have become the most important platform for online education [1]. Worldwide, various MOOC platforms such as Coursera and edX have emerged, extending access to courses from prestigious universities to millions of students. In China, XuetangX has risen as one of the largest MOOC platforms, offering thousands of courses and attracting a substantial user base [2]. Nevertheless, this proliferation of online courses and the constant evolution of knowledge pose a new challenge in the form of information overload for MOOC platforms [3–5]. Course recommendation [6–10] is an effective approach to tackling this challenge [11].

The formalized definition of course recommendation is: When presented with a set of historical courses that users enrolled in before time t, our goal is to recommend the most relevant courses that the user will enroll in at time t + 1 [12]. Clearly, it is particularly important to describe and model the user profile accurately for the recommendation model. To this end, the factor item similarity model (FISM) [13] can be used to extract implicit feedback from the learning behaviors of users, or the neural attentive item similarity (NAIS) [14] can be used to distinguish the importance of different historical courses to target courses by assigning different weight coefficients to historical courses. However, different from many other recommendation domains [15–19], as users' improvement in cognition level and interests change, they will enroll in diversified courses. Under these circumstances, the recommendation effect of courses with high contribution will be weakened by other courses with low contribution, which may result in recommending a course that does not align with the user's preferences. These low-contribution courses are called noise courses.

In order to solve the problem of noise courses, a hierarchical reinforcement learning (HRL) algorithm [12] can be used, acting on the user profile to remove noise courses in the historical course sequence using a two-level sequential decision process. The profile reviser model in the algorithm is then trained jointly with the recommendation model (NAIS).

While substantial advancements have been achieved in course recommendation using the HRL model, there are still some issues with the model. Specifically, the profile reviser of the model contains a high-level task and a low-level task; when the high-level task decides not to revise the user profile, even if there are some noise courses at this time, the model cannot remove them.

Inappropriate decisions will diminish the overall performance of the model and may converge to a local optimum state. For the low-level task, because the sampling of the policy function and the state transition function are both random probability values in the reinforcement learning (RL) method [20], removing some noise courses randomly will lead to instability of model prediction. When recommending target courses to users, some of the more relevant courses may be ignored. As shown in Figure 1, the HRL model recommended the target course "Monetary Finance" due to the noise course "Principles of Economics". But according to the historical courses, the user has studied "C++" and "Principles of Databases", so he or she should prefer the computer programming courses.

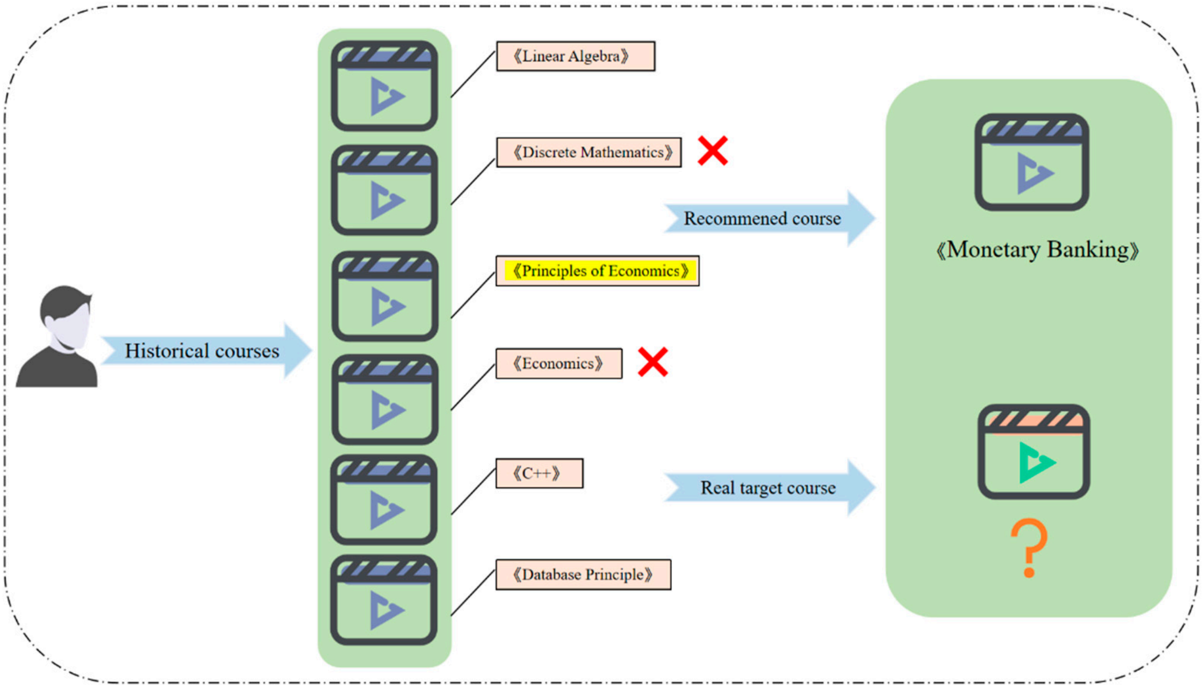


Figure 1. An example of HRL model for course recommendation.

In the above case, the user profile reviser does not guarantee the accuracy of decisions. In this work, we aim to reconstruct the profile reviser to improve the accuracy of course recommendation. To this end, we propose a new course recommendation model: the actor-critic hierarchical reinforcement learning (ACHRL) model. The actor-critic method (AC) embedded in the model can improve the accuracy of decision making efficiently in the process of revising the user profile. In particular, we adopt the policy gradient method [21] based on the temporal difference error [22]. This can reduce variance and improves robustness. The profile reviser can be updated in a timely manner, improving the accuracy of decisions adaptively. In summary, the contributions of our research are as follows:

- We propose an actor-critic hierarchical reinforcement learning model (ACHRL) to optimize the user profile reviser of the HRL model and improve the accuracy of course recommendations.
- We propose a policy gradient method based on the temporal difference error, which updates the policy adopted by the agent of the reinforcement learning model with rewards at the current moment and states of the moments before and after. This can speed up model convergence and improve the accuracy of the recommendation model.
- We performed experiments using two MOOC datasets, with the majority of users enrolled in courses from various categories, and our proposed model demonstrated a significant improvement over the baseline model on different evaluation metrics.

2. Related Work

2.1. Course Recommendation

The rapid development of Massive Open Online Course (MOOC) platforms has led to increased research and application of course recommendation systems. In general, course recommendations can be classified into four primary categories according to different methods:

Course Recommendation Based on Collaborative Filtering. Collaborative filtering methods encompass user-based and item-based collaborative filtering. These methods typically recommend courses to users based on the preferences of users with similar interests or the characteristics of courses. For example, Li et al. [23] designed a personalized online education platform based on collaborative filtering, resulting in more accurate and efficient course recommendations aligned with users' interests. **Course Recommendation Based on Content.** Content-based recommendation methods leverage information about the features of the courses to make recommendations. These methods are suitable for addressing cold-start problems as they do not rely on user interaction data. Ghauth et al. [24] proposed an e-learning recommendation system based on content-based filtering, which enhanced users' course outcomes by increasing the accuracy of the learning system. **Course Recommendation Based on Knowledge Graph.** Knowledge-graph-based recommendation methods utilize graph structures to represent the relationships between courses. They leverage domain knowledge and semantic relationships within the knowledge graph to model connections between a user's interests and courses, resulting in personalized recommendations. Xu et al. [25] applied knowledge-graph representation learning to embed semantic information of items into a low-dimensional semantic space and calculated the semantic similarity between recommended items. This approach enhanced recommendation performance at the semantic level. **Course Recommendations Based on a Hybrid Approach.** Hybrid recommendation methods combine multiple recommendation techniques, leveraging the strengths of each to provide a combined set of recommendations that aim to enhance accuracy and diversity. Hybrid approaches have become a popular solution for addressing various recommendation challenges simultaneously, offering greater adaptability compared to single-method approaches. Emon et al. [26] used a hybrid approach that combined association rule mining and user-based collaborative filtering to develop a recommendation system that identified the unique interests of different students in learning materials more effectively, resulting in more personalized recommendations. Gao et al. [27] proposed a personalized course recommendation model based on a convolutional neural network combined with negative sequence pattern mining. The model models the course-learning sequence as a negative sequence pattern according to the user's course registration, degree of completion, and final grades, and then via a convolutional sequence-embedding model. Each user can be recommended a course list. The model has achieved a good recommendation effect.

With the development of deep learning, the latest course recommendation model mainly adopts deep learning technology. Wang et al. [28] have developed a hyperedge-based graph neural network. The authors innovatively treat learners as the sets of courses in a hypergraph. The hyperedge-based graph attention network is proposed. The model provides accurate recommendation results. Ren et al. [29] have proposed a deep course recommendation model with multimodal feature extraction based on the long- and short-term memory network (LSTM) and attention mechanism. The paper makes use of the multimodal technique for building a complete learner portrait to recommend the courses that fit the user's preference.

2.2. RL-Based Course Recommendation

In recent years, there have been significant breakthroughs in reinforcement learning (RL) within the field of recommendation systems [30]. RL has achieved remarkable success in various domains, including e-commerce, video, and gaming [31–33]. These advancements have also opened up new opportunities in the realm of course recommendations within the education sector. As an interactive recommendation approach, RL-based recommendation models can continuously update their recommendation policies by receiving real-time feedback from users. This dynamic approach aligns more closely with real-world recommendation scenarios compared to traditional static methods. Particularly, deep reinforcement learning (DRL) [34–37] stands out for its capacity to process large-scale data, extract underlying features, and accurately achieve specific goals via end-to-end learning. One of the challenges in traditional reinforcement learning methods [38–40] is the issue of the "dimensional disaster". When the environment is complex or the task is intricate, the state space of the agent becomes too extensive. This results in a rapid increase in the number of parameters to be learned and the memory space required. Consequently, achieving the desired results becomes difficult. To address the dimensional disaster problem, researchers have introduced hierarchical reinforcement learning (HRL) [41–43]. The objective is to break down complex problems into smaller, more manageable subproblems and solve the original task by addressing these subproblems individually. In complex recommendation tasks, optimizing the policy directly according to the final goal can be inefficient. Hierarchical methods offer an effective means of enhancing recommendation efficiency by breaking down complex tasks into more manageable components.

HRL has made great progress in MOOC course recommendation. The accurate construction of the user profile model is the foundation of a recommendation system. Zhang et al. [12] applied HRL technology to course recommendation for the first time. The authors believed that the noise courses in the sequence of historical courses would affect the weight of the courses with real contributions. The noise courses are removed using the reinforcement learning method. Utilizing the revised data can indeed enhance the accuracy of course recommendations. On the basis of the above work, Lin et al. [44] proposed a hierarchical reinforcement learning model with dynamic recurrent mechanism for course recommendation, and it employed a recurrent scheme by context-aware learning to exploit the current knowledge. Lin et al. [45] focused on improving the performance of the profile reviser, making the agent obtain the cumulative rewards of the context and adopt better policies. Lin et al. [46] used a recommendation model that could capture user preferences by historical courses and improve the accuracy of course recommendations. Inspired by the above methods, we propose the ACHRL model to optimize the profile reviser, which ensures more accurate decisions on whether to delete noise courses. Finally, the accuracy of the recommendation results is improved.

3. Preliminaries

Reinforcement learning is a sequential decision-making process, and reinforcement learning models need to find an optimal policy to optimize the target. In this section, we mainly introduce the methods for obtaining the optimal policy using reinforcement learning, the policy gradient method, and the actor-critic method.

3.1. Policy Gradient Method

We need to learn a target policy explicitly to define the policy-based method [47] in reinforcement learning. The policy gradient is the basis of the policy-based method. It can parameterize the policy. Our objective is to discover an optimal policy that maximizes the expected return in the given environment. Therefore, the update of the objective function approximates the gradient ascent in $J(\theta)$ as follows:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)} \tag{1}$$

where θ_t is the policy parameter at time t, α is the learning rate which controls the step size of parameter updates, $J(\theta_t)$ is a random estimate, and its expected value can be approximated as the gradient of $J(\theta_t)$ according to its parameter θ_t. The method that follows Equation (1) is referred to as the policy gradient method.

3.2. Actor-Critic Method

In reinforcement learning, the policy-based method needs to learn a policy function, and the value-based method needs to learn a value function explicitly. The actor-critic (AC) method [48] combines both aspects, enabling the simultaneous learning of both the policy function and the value function. This combination allows the value function to assist in the more effective learning of the policy function. In the context of policy gradients, we can express the gradient formula in a more generalized form as follows:

$$\begin{gathered} g = \mathbb{E}\left[\sum_{t=0}^{T} \psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \\ \psi_t = r_t + \gamma\, V_\omega(s_{t+1}) - V_\omega\,(s_t) \end{gathered} \tag{2}$$

where $\mathbb{E}$ represents the expected value of a random variable based on the policy function π. In this paper, ψ_t is the temporal difference error to guide the learning of the policy function. $\pi_\theta(a_t \mid s_t)$ represents the policy function; we could call it "Actor".

In this paper, we employ the actor-critic method, as illustrated in Figure 2. The value function in this method plays a crucial role in guiding the learning of the policy function and optimizing the agent's action selection. We update the value function using the temporal difference error (TD error). Notably, the actor-critic method is essentially a policy-based approach, because its goal is also to optimize a policy with parameters. In summary, we utilize a policy gradient method based on the temporal difference error to enhance the objective function.

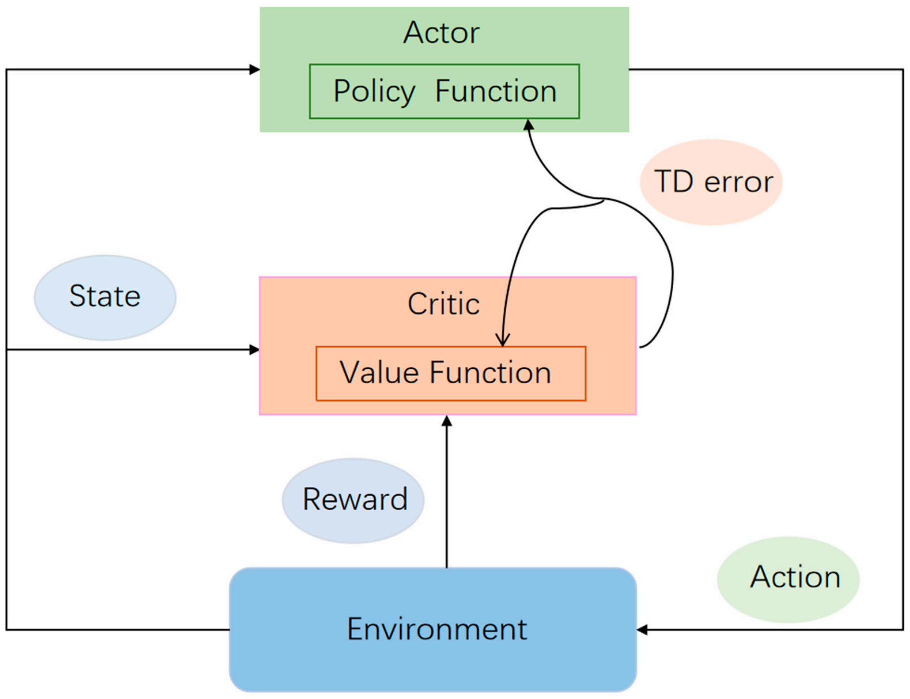


Figure 2. The execution process of the AC method.

4. Methodologies

4.1. Definitions and Problem Formalization

We first define some concepts as the basis of later discussion.

Definition 1. *The sequence of historical courses is denoted as ε^u, and the elements in the sequence represent one course per user u enrolment $\varepsilon^u = \left(e_1^u\ e_2^u\ e_3^u \ldots e_t^u\right)$.*

Definition 2. *A course embedding vector is denoted as p_t^u and aggregates the embeddings of all the historical courses: $\mathrm{p}_1^u \ldots \mathrm{p}_t^u$.*

Definition 3. *The recommendation probability is denoted as $\rho_{u,i}$. It is the result of an operation on the course embedding vector $\rho_{u,i} = P(y = 1 \mid \hat{\varepsilon}^u, c_i)$, where c_i denotes the target courses recommended to users. $\hat{\varepsilon}^u$ represents a modified ε^u sequence.*

The problem to be solved in this paper is formulated as follows. Given the user's historical enrolled course sequence ε^u of the online platform, we build a profile reviser under the framework of hierarchical reinforcement learning from which we can obtain the course embedding vectors $\mathrm{p}_1^u \ldots \mathrm{p}_t^u$ in the modified course sequence $\hat{\varepsilon}^u$, and then calculate a recommendation probability value $\rho_{u,i}(0,1)$ of these vectors in the attention mechanism-based recommendation network. We measure the validity of the data recommendation in the current course list by comparing the probability value with a set threshold value, and then guide the operation of the profile reviser. The recommendation module and the profile reviser module will form a joint training.

Next, we provide an analysis and presentation of the methodology of the proposed model.

We conduct qualitative research using MOOCCourse and MOOCCube datasets. Because the two datasets are publicly available online datasets and contain rich information, such as the time the user registered for the course, the type of course, the name of the course, and the index of the course, we can conveniently convert the data into implicit feedback represented by 0 or 1 when conducting the experiments. We collected data from XuetangX, one of the largest MOOC platforms in China. It is all real-user data with most users enrolled in multiple courses. We mainly analyzed the impact of the number of courses registered by users on the recommendation results in the data. Our method is the most common general method in the field of recommendation. We take the historical data as the training set and verify the experimental results using the experimental set. We take the last course in the user registration course as the recommended target course to verify the recommendation effect of the model. Of course, the method has some limitations; because the method we adopt requires a longer sequence of historical data to have better recommendation results, it is more friendly to long-sequence datasets.

4.2. Framework

Overview. The ACHRL model is depicted in Figure 3. It consists of two parts: a profile reviser module optimized using the AC method and a recommendation module based on attention mechanism. The profile reviser is a hierarchical reinforcement learning framework consisting of a two-layer Markov decision process (MDP). In this framework, the high-level task is responsible for determining whether to revise the user profile, while the low-level task is tasked with deciding which noise course should be removed. When the high-level task determines to revise the user profile, the delay reward can be obtained after removing the last noise course in the low-level task. Two temporal difference errors from the AC method, TD-ERROR_H and TD-ERROR_L, can guide the updating of high-level policy and low-level policy, respectively. After completing the task of the profile reviser, the embedding vector of the courses is fed into the recommendation module generating the recommended probability value. In the ACHRL model, we optimize the two-layer tasks of the profile reviser so that the agent (a^h, a^l) can take more accurate actions. This can provide more accurate data for recommendation modules (the variables involved in the figure are explained in the introduction to the two modules of the model).

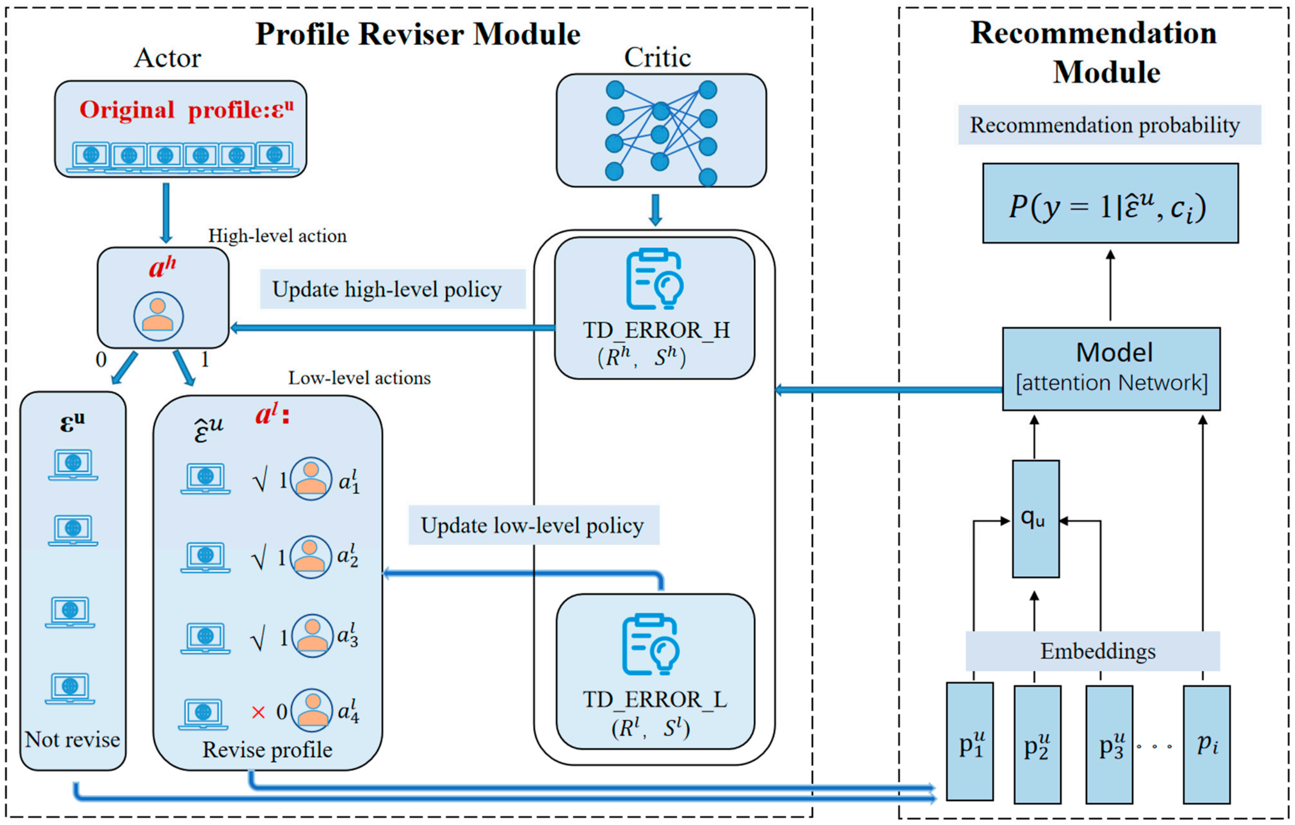


Figure 3. The overall framework of the proposed model (ACHRL).

4.2.1. Profile Reviser Module

In the profile reviser, the agent selects the high-level action decision according to the high-level policy and removes the noise course according to the low-level policy. The key challenge is how the two-level task can revise the user profile more accurately. We use the AC method to optimize the HRL model to revise the user profile. Next, we will introduce the specific composition of the profile reviser based on HRL and its optimization using the AC method in detail.

HRL for the profile reviser module. The modification task of the user profile can be divided into a two-layer Markov decision process (MDP) [49]. The first layer is a high-level task that decides whether to revise the user profile, and the second layer is a low-level task responsible for determining which historical course to remove. They can be transformed into a 5-tuple Markov decision process (MDP) denoted as $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \beta \rangle$.$\mathcal{S}$ represents a set of states, and $\mathcal{A}$ represents a set of actions. $\mathcal{P}$ is the state transition function. $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ converts to a probability value of [0, 1] by mapping. $\mathcal{R}$ is a function used to represent rewards. $\beta \in \{0,1\}$ signifies the discount factor applied to the reward at each moment. The following paragraph describes the content from [12].

State. In the high-level task, we define the state features s^h, which is characterized as the average cosine similarity and the average element-wise product between the embedding vectors [50] of each historical course e_t^u and the target course c_i. In the low-level task, we define the state feature s_t^l, which is characterized as the cosine similarity [51], the element-wise product, and the average of the two previous features between the embedding vector of the current historical course e_t^u and the target course c_i.

Action. Action is a discrete variable. The action $a^h \in \{0,1\}$ in the high-level task determines whether to revise the profile of a user or not, and the action $a^l \in \{0,1\}$ in the low-level task represents whether to remove the historical course e_t^u or not.

Reward. Reward is used to indicate whether the performed actions are reasonable or not. When deciding to revise the user profile, the reward can be obtained after the last action is executed in the low-level task. Before that, the reward is zero. Here, $R_h\left(s_t^h, a_t^h\right)$ is defined as the difference between the log-likelihood after and before the profile is revised. The reward for the high-level task is defined as follows:

$$\begin{aligned} R_h\left(s_t^h, a_t^h\right) &= \log P(\hat{\varepsilon}^u, c_i) \\ &\quad -\log P(\varepsilon^u, c_i) \quad if\ t = t_u \\ R_h\left(s_t^h, a_t^h\right) &= 0 \qquad\qquad\qquad if\ t \neq t_u \end{aligned} \tag{3}$$

where s_t^h denotes the state of the high-level task at time t, a_t^h denotes the action of the high-level task at time t, and $\hat{\varepsilon}^u$ is the revised profile, which is a subset of ε^u. The ε^u is the original profile. c_i denotes a target course, and $P(\hat{\varepsilon}^u, c_i)$ is an abbreviation of $P(y = 1 \mid \hat{\varepsilon}^u, c_i)$. The logarithmic function is used in the formula for better convergence for the training of the model.

For the reward $R_l\left(s_t^l, a_t^l\right)$ in the lower-level task, aside from the common component shared with the high-level task reward, there is an internal reward in the low-level task, which is defined as the difference of the average cosine similarity between each historical course e_t^u after and before the user profile is revised and the target course c_i. Internal reward is used to optimize the efficiency of the policy execution of the agent in the low-level task. The reward of the low-level task is defined as follows:

$$\begin{aligned} R_l\left(s_t^l, a_t^l\right) &= (\log P(\hat{\varepsilon}^u, c_i) - \log P(\varepsilon^u, c_i)) \\ &\quad + \left(\frac{\sum_{t=1}^{\hat{t}_u}\left(e_t^{uT} c_i\right)}{\hat{t}_u} - \frac{\sum_{t=1}^{t_u}\left(e_t^{uT} c_i\right)}{t_u}\right) \quad if\ t = t_u \\ R_l\left(s_t^l, a_t^l\right) &= 0 \qquad\qquad\qquad if\ t \neq t_u \end{aligned} \tag{4}$$

where s_t^l represents the state of the low-level task at time t, a_t^l represents the action of the low-level task at time t, and $e_t^{uT} c_i$ represents the cosine similarity between the historical courses e_t^u and the target course c_i.

Policy. The policy function of the high-level task is:

$$\begin{gathered} \pi_\theta\left(s_t^h, a_t^h\right) = \mathrm{softmax}\left(a_t^h \sigma\left(W_2^h z_t^h\right) + \left(1 - a_t^h\right)\left(1 - \sigma\left(W_2^h z_t^h\right)\right)\right) \\ z_t^h = \mathrm{ReLU}\left(W_1^h s_t^h + b^h\right) \end{gathered} \tag{5}$$

where $W_1^h \in \mathbb{R}^{d_1^h \times d_2^h}$, $W_2^h \in \mathbb{R}^{d_2^h \times 1}$ and bias vector $b^h \in \mathbb{R}^{d_2^l}$ are the parameters to be learned in the neural network, d_1 is the size of the hidden layer,d_2 is the number of state features, z_t^h is the size of the hidden layer of the neural network at time t in the high-level task, and σ is the non-linear activation function used to transform the state into a probability value. ReLU is an activation function. Therefore, the policy parameter of the high-level task can be defined as follows $\theta^h = \left\{W_1^h, W_2^h, b^h\right\}$. Similarly, we can obtain the low-level task policy parameter as $\theta^l = \left\{W_1^l, W_2^l, b^l\right\}$.

The AC method optimizes the profile reviser. Different from the decision made by the agent relying on the policy function, the action selection of the agent in the AC method has higher accuracy. Based on this, we use the AC method in the hierarchical task of the profile reviser. In this case, instead of relying on the cumulative rewards obtained in each training round for policy updates, the agent performs policy updates via the temporal differential error $\delta(\mathrm{t})$. The $\delta(\mathrm{t})$ is estimated using the critic network of the AC method. In the calculation process of $\delta(\mathrm{t})$, we use the rewards of the current moment and the states of the moments before and after. It is more instructive for the policy update. The formula is as follows:

$$\delta(\mathrm{t}) = r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t) \tag{6}$$

where V_ω represents the state value network in reinforcement learning, ω is the parameter of the value network, s_t and s_{t+1} represent the state of the current moment and the next moment, respectively, and r_t is the reward of the current moment. $\gamma \in (0,1)$ is a discount factor for cumulative rewards of the agent.

4.2.2. Policy Gradient Based on Temporal Differential Error

The previous section Algorithm 1 illustrates that the value function in the AC method can guide the policy function to enable the agent to make more accurate decisions. We use the temporal difference error in the gradient update process, and the gradient formula is as follows:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \nabla_\theta \pi(a \mid s, \theta)(r_t \\ +\gamma V_\omega(s_{t+1}) - V_\omega(s_t)) \tag{7}$$

The $\propto$ in the formula represents that the two sides of the formula are proportional, $\mu(s)$ represents a policy distribution in the policy function, and $r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t)$ is the temporal differential error $(\delta(\mathrm{t}))$.

Algorithm 1 Actor-Critic Method

```
Input: a derivable policy parameterization $\pi_\theta(s,a)$, a derivable actor-value, a derivable
parameterization $Q^{\pi_\theta}(s_t,a_t)$, a state-value parameterization $V_\omega(s_t)$
Initialize: parameters $\theta = \theta^0$, $W = W^0$
1: for sequence $e = 1$ to $E$ do :
2:         Generate a sampling sequence {s1,a1,r1,s2,a2,r2...}          following $\pi_\theta$
3:         for data at each step do
4:         $\delta_t = r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t)$
5:     $w = w + \alpha_\omega \sum_t \delta_t \nabla_\omega V_\omega(s_t)$
6:     $\theta = \theta + \alpha_\theta \sum_t \delta_t \nabla_\theta \log \pi_\theta(s_t,\ a_t)$
7:     end for
```

4.2.3. Objective Function

The ACHRL model improves the accuracy of course recommendations. For the hierarchical tasks, the temporal difference error is used in the policy gradient method to optimize policy function. It is calculated using the value of rewards and states. For our model, the objective function is to maximize the expectation of expected rewards:

$$\theta^* = \arg\underset{\theta}{max} \sum_\tau P_\theta(\tau) T(\tau) \tag{8}$$

The θ represents the policy parameter: θ^h or t θ^l. τ represents the sampling sequence of the actions and transition states. For the high-level task, it can be represented as $\{s^h,\ a^h\}$, and for the low-level task, it can be represented as $\{s_1^l, a_1^l \ldots s_t^l, a_t^l\}$. $P_\theta(\tau)$ is the probability of the state transition in the high-level task or the low-level task, and $T(\tau)$ is the temporal differential error $\delta(\mathrm{t})$.

The policy function for the high-level task is as follows:

$$\nabla_\theta = \frac{1}{n} \sum_{n=1}^{N} \sum_{t=1}^{t_u} \nabla_\theta \log \pi_\theta(s^n, a^n) r_t^h \\ +\gamma V_\omega^h(s_{t+1}) - V_\omega^h(s_t) \tag{9}$$

where t_u represents the quantity of historical courses. r_t^h is the real reward obtained by the high-level task at time t. V_ω^h is the representation of the state value network V_ω in the high-level task, and ω is the updated parameter in the state value network $V \cdot r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t)$ is the temporal difference error.

In the low-level task, the delayed reward can be obtained after removing the last noise course. The temporal difference error can be obtained via the calculation of the rewards and states. The policy function of the low-level task is as follows:

$$\nabla_\theta = \frac{1}{n}\sum_{n=1}^{N}\sum_{t=1}^{t_u}\nabla_\theta \log \pi_\theta(s_t^n, a_t^n) r_t^l \\ +\gamma V_\omega^l(s_{t+1}) - V_\omega^l(s_t) \tag{10}$$

where r_t^l is the real reward obtained at time t of the low-level task. V_ω^l is the representation of the state value network V_ω in the low-level task. The meanings of other terms in the formula can be analogous to the relevant definitions in the above high-level policy network.

4.3. Attention-Based Recommendation Module

In our model, the recommendation module is the NAIS model [14]. It is a recommendation model that relies on an attention mechanism, and it excels at efficiently converting the course embedding vectors obtained from the profile reviser into recommended probability values. Besides this, it can provide a fair and reasonable verification for our ACHRL model and the HRL model. The following is a basic introduction of the NAIS model.

$$\mathrm{q}_u = \sum_{t=1}^{t_u} a_{it}^u \mathrm{p}_t^u, a_{it}^u = f(\mathrm{p}_t^u, \mathrm{p}_i) \tag{11}$$

where q_u represents the embedding vector of the profile, and p_t^u is an embedding vector of the historical courses e_t^u. p_i is an embedding vector of the target course c_i, and a_{it}^u is an attention weight coefficient of the historical course e_t^u during the recommendation process. $f(\mathrm{p}_t^u, \mathrm{p}_i)$ is an attention function. Its essence is one multi-layer perceptron (MLP) [52].

$$f(\mathrm{p}_t^u, \mathrm{p}_i) = \mathrm{h}^\top \mathrm{ReLU}\left(\mathrm{W}^{at}(\mathrm{p}_t^u \odot \mathrm{p}_i) + \mathrm{b}^{at}\right) \tag{12}$$

where $\mathrm{h}^\top$ is the attention weight vector mapped by the hidden layer of the neural network, W^{at} is the weight matrix, and b^{at} is a bias vector. ReLU is an activation function.

According to $\mathrm{q_u}$ and p_i, the attention-based recommendation module generates a recommendation probability $\rho_{u,i}$.

$$\rho_{u,i} = P(y = 1 \mid \hat{\varepsilon}^u, c_i) = \sigma\left(\mathrm{p}_i^T \mathrm{q}_u\right) \tag{13}$$

If the value of y is 1, the recommendation is successful, and σ is an activation function used to convert the input embedding to the probability value in the recommendation model.

4.4. Separable Two Components of ACHRL

The model proposed in this paper is called ACHRL to indicate that we optimize the two-layer structure of the profile reviser by using the AC method. We use the term ACHRL_H to represent the optimization of the high-level task. The term ACHRL_L represents the optimization of the low-level task. Next, we will provide a comprehensive description of both of them. In the ablation experiment, we will compare both model methods with HRL and ACHRL.

4.4.1. High-Level Task Optimization: ACHRL_H

As has been introduced above, a major problem of the HRL model is that the action of the agent is random. The high-level task cannot decide whether to revise the user profile accurately. The ACHRL_H model enhances the precision of decision-making for the high-level task. To ensure a fair performance comparison with the HRL model, we employ the REINFORCE algorithm [53] to calculate the gradient of the low-level task policy function too:

$$\nabla_\theta = \frac{1}{n}\sum_{n=1}^{N}\sum_{t=1}^{t_u}\nabla_\theta \log \pi_\theta(s_t^n, a_t^n) R_l(s_t^n, a_t^n) \tag{14}$$

where $R_l(s_t^n, a_t^n)$ represents the cumulative delay reward of the low-level task obtained by each sampling sequence τ.

4.4.2. Low-Level Task Optimization: ACHRL_L

In the HRL model, the key to the low-level task is how to identify the noise course and remove it accurately. Due to the limitation of the policy function itself, the low-level task removes noise courses with randomness, which makes the execution of the policy less accurate, and may even remove the valid course mistakenly, affecting the recommendation performance. To solve this problem, we use ACHRL_L to assist the low-level task in removing noise courses for improving the accuracy of low-level action selection. While removing noise courses, we reserve the target course for each user as far as possible. We still use the REINFORCE algorithm to calculate the gradient of the high-level task policy function:

$$\nabla_\theta = \frac{1}{n}\sum_{n=1}^{N}\sum_{t=1}^{t_u}\nabla_\theta \log \pi_\theta(s^n, a^n) R_h(s_t^n, a_t^n) \tag{15}$$

where $R_h(s_t^n, a_t^n)$ represents the cumulative delay reward for each sampling sequence in the high-level task. Note here: Some element representations in Formulas (9), (10), (14), and (15) are simplified, and we omit h and l representing high and low levels.

4.5. Model Training

The previous section outlines that the entire model can be split into two components. The training process involves three key steps: initially, pre-training the recommendation module [54], followed by pre-training the profiler reviser module, and finally, jointly training the two models together.

Next, we illustrate the details of its interaction by combining Figure 4 and Algorithm 2. First, we pre-train our recommendation module (i.e., $\varnothing^0 = \{\mathrm{h}^\top, \mathrm{W}^{at}, \mathrm{b}^{at}\}$) using the original dataset (ε^u). Then, we conduct the pre-training of the profile reviser module (i.e., $\theta^0 = \{w_1, w_2, b\}$). In this process, the update of the high-level policy and the low-level policy depend on the optimization of the temporal difference error (TD-ERROR_H and TD-ERROR_L) calculated using the AC method. Finally, we conduct the joint training of the two models. If the output $P(y = 1 \mid \hat{\varepsilon}^u, c_i)$ of the recommendation model is greater than the threshold set in the experiment (e.g., 0.6), the recommendation model has obtained a reasonable recommendation result. At this time, the high-level task makes the decision corresponding to "0" in Figure 4, which means that the user's profile is not modified. No reward will be given in this case and the δ^h is zero (corresponding to the fifth line in Algorithm 2). The course embedding vector of the unrevised profile is input into the recommendation model directly. On the contrary, the high-level task takes the decision corresponding to "1" in Figure 4, which means that revision of the user's profile is required. At this time, noise courses need to be removed. In the low-level task, "0" means remove the course and "1" means keep the course. When the low-level task is completed, both the high-level task and the low-level task obtain the reward. The corresponding values are also obtained for both δ^h and δ_t^l. Under the optimization of them, the removing work of all noise courses is completed, and the user profile is revised (lines 7–9 of Algorithm 2). The revised profile is input into the recommendation model as an embedding vector. The next round of joint training begins until the end of the training. The parameters are updated during the training process (lines 12–13 of Algorithm 2).

Algorithm 2 Actor-Critic-Based Hierarchical Reinforcement Learning

```
Input: training data: $\varepsilon^u$; pre-train recommendation model parameterized by:
$\varnothing^0 = \{h^\top, W^{at}, b^{at}\}$; pre-train profile reviser parameterized by: $\theta^0 = \{w_1, w_2, b\}$; derivable state
value functions: $v_h(s,w)$, $v_l(s,w)$
Initialize: $\varnothing = \varnothing^0$, $\theta = \theta^0$, $\delta^h(r,s) = 0$, $\delta^l(r,s) = 0$;
1:  for sequence k = 1 to k do
2:  for each $\varepsilon^u$: = $(e^u{}_1, \ldots e^u_{t_u})$ and $c_i$ do
3:   Sample a high-level action $a^h$ with $\theta^h$, in the high-level task;
4:     if $P(y = 1 \mid \hat{\varepsilon}^u, c_i) > 0.6$ then
5:          $\delta^k_h\left(r^h_t, s^h_t\right) = 0$
6:   else
7:            Sample a sequence of states and actions with $\theta^l$, in the low-level task;
8:         Calculate: $\left(r^h_t, s^h_t\right)$, $\delta^k_l\left(r^h_t, s^h_t\right)$;
9:         Calculate the gradients using Equations (9) and (10) according to Algorithm 1;
10:     end if
11:     end for
12:      Update parameters $\theta^h, \theta^l, w^h, w^l$ by the gradients;
13:      Update parameter $\varnothing$ by the recommendation module;
14: Output the recommendation probability $P(y = 1 \mid \hat{\varepsilon}^u, c_i)$ using Equation (13)
15:     end for
```

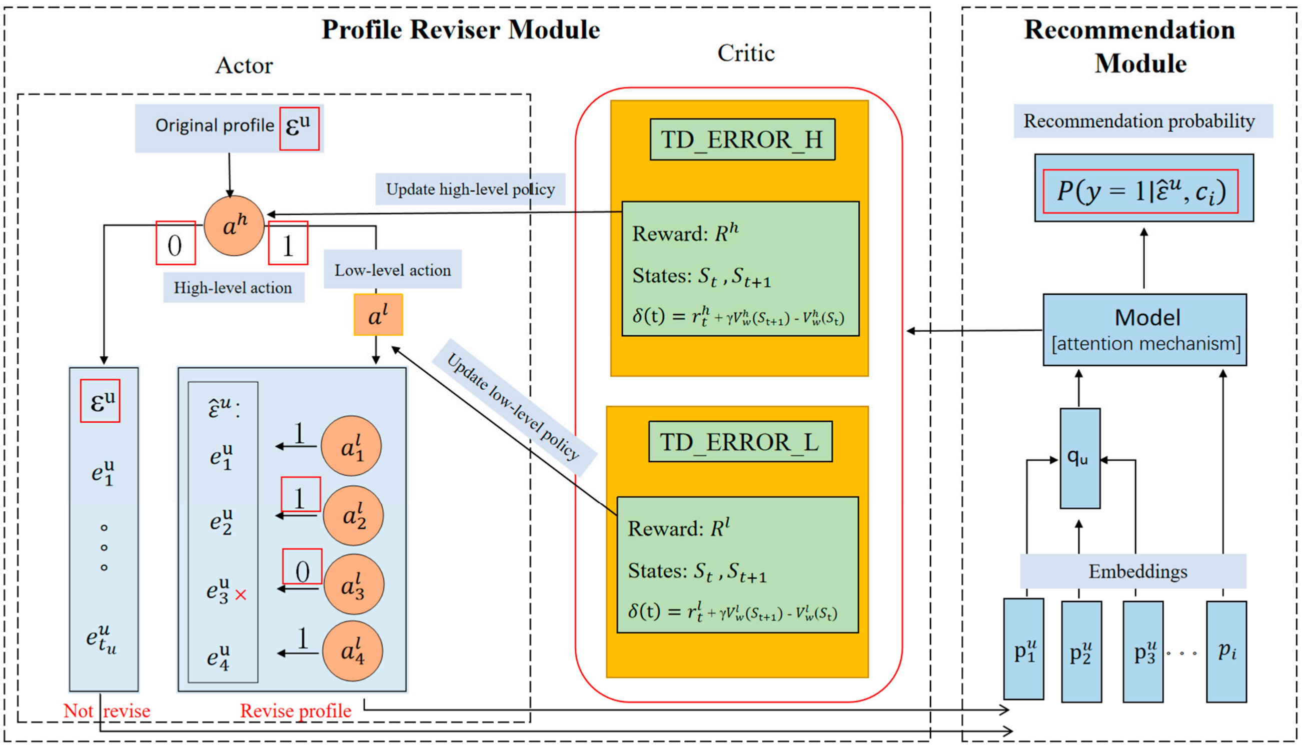


Figure 4. Interaction details of recommendation module and profile reviser.

5. Experiments

In this section, we will begin by introducing two datasets and explaining the experimental setup. Subsequently, we will delve into the results and findings of the experiment.

5.1. Dataset

We conduct the experiment using the real-world datasets MOOCCourse (http://moocdata.cn/data/course-recommendation (accessed on 8 May 2023)) and MOOCCube. Both of these datasets are from the domestic MOOC platform “XuetangX” (http://www.xuetangx.com (accessed on 5 February 2023)). Table 1 shows the details of the two datasets.

Each user registered for at least three courses in the MOOCCourse dataset, and each user registered for at least two courses in the MOOCCube dataset. To ensure the fairness of the experiment, we adopted the same data preprocessing method as the HRL mode [12]. We structured the user enrollment behavior in such a way that the actions in the training phase were taken in the training set before the test set. Each entry in both training and test sets includes a series of previous courses alongside the user's target courses. For training, we identified the final course in each sequence as the target course, with all the previous courses treated as historical. In the testing phase, every historical course in the test set was regarded as the target course, and the corresponding course from the training set for the same user was considered the historical course.

Table 1. The experimental datasets.

Dataset	Courses	Users	Interactions	Average Interactions
MOOCCourse	1302	82,535	458,453	5.55
MOOCCube	706	55,203	190,049	3.44

5.2. Experimental Setup

5.2.1. Compared Method

To evaluate model performance, we compared the ACHRL model with some baseline models as follows:

- **MLP [52]**: A powerful recommendation system model that leverages deep learning techniques to provide personalized recommendations based on complex user–item interactions.
- **FISM [13]**: An important model in the field of recommendation systems, particularly suitable for collaborative filtering recommendation problems where user historical behavior data is the primary input.
- **NeuMF [55]**: A model that combines matrix decomposition techniques and MLP methods to mine the potential information of user courses for modelling and recommends relevant courses to users.
- **NARM [56]:** An optimized gated recursive model that evaluates the attention factor based on the behavior and primary purpose of users.
- **NAIS [14]**: The model built using an item-based collaborative filtering method combined with an attention mechanism neural network that can distinguish between different historical course weights for course recommendation.
- **HRRL [45]**: An HRL-based method using time-context rewards can optimize strategy learning in reinforcement learning for course recommendation.
- **DARL [46]**: A novel course recommendation framework that can capture user preferences using historical data for improving the effectiveness of course recommendations.

We investigate the recommendation performance of the HRL model and two variants of the ACHRL model (ACHRL_H and ACHRL_L).

- HRL [12]: Recommendation model and profile reviser joint training.
- ACHRL_H: A simplified version of the ACHRL model and profile reviser joint training that only adopts the optimization of the AC method in the high-level task of the profile reviser.
- ACHRL_L: A simplified version of the ACHRL model and profile reviser joint training that only adopts the optimization of the AC method in the low-level task of the profile reviser.

5.2.2. Evaluation Metrics

We employ the most authoritative evaluation metrics used in the recommended field [57]:

- HR@K (Hit Ratio at K): A measure of how many of the relevant items were successfully included in the top K recommendations. The following formula represents a successful recommendation to the user

$$HR@K = \frac{\sum_{u=1}^{U} H_{its_u@K}}{|GT|} \tag{16}$$

  where GT represents the total number of items corresponding to users in all test sets, and $H_{its_u@K}$ represents the count of items that users interact with or find relevant in the previous K recommendations.
- NDCG@K (Normalized Discounted Cumulative Gain): An accumulative performance measure that takes into account both the relevance and position of ranked items. It can be defined as follows:

$$\begin{aligned} \text{NDCG@K} &= \frac{1}{U}\sum_{u=1}^{U} \frac{DCG_u@K}{IDCG_u@K} \\ DCG_u@K &= \sum_{i=1}^{K} \frac{2^{\text{reli}}-1}{\log_2(i+1)} \end{aligned} \tag{17}$$

  where $IDCG_u@K$ represents the discount cumulative gain of the optimal TOP-K recommendation list, and "*reli* " represents a measure of the correlation between the recommended result and the target.

5.2.3. Parameters and Environment

Experimental Environment: We implement the model using Tensorflow2 and deploy it using a Linux server equipped with an NVIDIA RTX 3090 GPU featuring 20 GB of video memory.

Parameter Settings: Recommendation module: The size of both the course embedding vector D_e and the hidden layer D_h are configured as 16. The batch size is 256. The learning rate is 0.02. Profile reviser: The sampling time N is set to 3, and the learning rate for the pre-training model and the joint training model is set to 0.001 and 0.0005. For the policy network, the hidden layer ($d^h{}_1$,$d^h{}_2$) is configured as 8, the state size ($d^h{}_2$) of the high-level layer is 18, and the state size ($d^l{}_2$) of the low-level layer is 34. The discount factor of reward β is 0.5. ACHRL model parameter: The sampling time N is also 3, and the hiding layer size of the recommendation module is configured as 16. For the value network, the hidden layer size ($d^h{}_1$,$d^h{}_2$) is 20. The state ($d^h{}_2$) size of the high-level layer is 18. The state ($d^l{}_2$) size of the low-level layer is configured as 34.

5.3. Results and Analysis

5.3.1. Comparison of Experimental Results

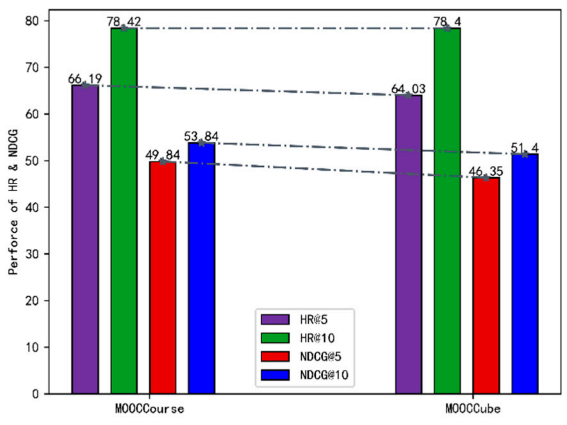
Table 2 demonstrates that our three proposed models achieve better results in terms of recommendation performance compared to the baseline models. For the MOOCCourse dataset, the HR was improved from 0.13% to 6.35% and the NDCG was improved from 0.59% to 5.34%. For the MOOCCube dataset, the HR was improved from 5.58% to 6.35% and the NDCG was improved from 1.48% to 2.12%. In summary, the above results provide a proof of the effectiveness of the ACHRL model. This means that our model can provide effective guidance for users in course selection. For online platforms, the accuracy of the model recommendation can bring a better experience to users and further strengthen the popularity of online platforms.

In Table 2, of all the baseline models, namely NeuMF, FISM, and MLP, they exhibit the worst performance. This is primarily due to their inability to account for the contribution of various historical courses when making recommendations for target courses. NARM and NAIS underperform in comparison to all HRL-based models. This performance gap arises from their limited ability to differentiate the impact of historical courses when users are enrolled in a larger variety of courses. All models based on HRL surpassed the performance of other baseline models. This is because the HRL-based models optimize the user profile. It makes the updated data represent the preferences of users more accurately and improves the accuracy of the recommendation. This means that, in scenarios like course recommendations, our model accurately reflects user preferences and may be more generalizing. It should be noted that platforms similar to online courses, such as music platforms and broadcasting platforms, have strong consistency in the recommended data objects, which is a factor that must be considered in determining the promotional scope of our model.

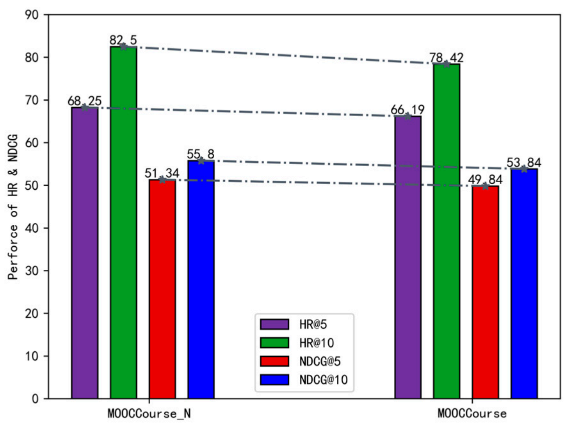
Table 2. Recommendation performance (%).

	MOOCCourse				**MOOCCube**			
	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10
MLP	52.53	66.74	40.61	40.96	51.62	66.55	40.00	43.58
FISM	53.12	65.89	40.63	45.13	52.85	65.80	40.50	45.52
NeuMF	54.20	67.25	42.06	46.05	54.25	67.50	41.72	46.00
NARM	54.23	69.37	42.54	47.24	54.12	69.50	41.85	47.20
NAIS	56.05	68.98	43.58	47.69	56.02	69.53	43.50	47.23
HRL	59.84	75.00	44.50	50.95	58.45	72.05	44.87	49.28
HRRL	61.36	78.29	45.82	51.70	-	-	-	-
DARL	63.12	77.63	48.53	53.25	-	-	-	-
ACHRL_H	63.61	77.21	46.07	51.18	61.95	76.98	45.42	50.67
ACHRL_L	64.96	78.04	48.58	52.86	62.91	77.13	46.33	51.07
ACHRL	**66.19**	**78.42**	**49.84**	**53.84**	**64.03**	**78.40**	**46.35**	**51.40**

In particular, we construct a MOOCCourse_N dataset based on the MOOCCourse dataset, which has 82,535 users and 1302 courses, with each user enrolled in at least 5 courses, and an average of 9.25 courses per user. Comparing the performance of the ACHRL model on the different datasets in Figure 5, it is clear that the ACHRL model performs better on the MOOCCourse dataset and MOOCCourse_N dataset, respectively. This confirms that the ACHRL model has a good recommendation effect to those users who are interested in multiple courses. More courses help the model to train adequately and to remove noise courses accurately. This can recommend target courses to users more accurately.



(**a**) MOOCCourse and MOOCCube dataset



(**b**) MOOCCourse_N and MOOCCourse dataset

Figure 5. Recommendation performance for HR and NDCG (%) of ACHRL model.

In addition, to further verify the effectiveness of the ACHRL model, we compare it with the two newest models: HRRL and DARL in the past two years, and the ACHRL model exhibits the highest performance. Since the MOOCCube dataset of this thesis is slightly different from the MOOCCube dataset used by the two models mentioned above, the experimental data of the comparison is not given in the corresponding position in Table 2. In Table 3, we describe the key parameters used for the existing two newest models to ensure a fair comparison, and what we mention above in "Parameter Settings" is consistent with the data in the table.

Table 3. Model parameters of HRRL and DARL.

Recommendation Module	**Course Embedding Size**	**Course Hidden Layer Size**	**Batch Size**	**Learning Rate**
HRRL	16	16	256	0.02
DARL	16	16	256	0.02
profile reviser module:	sampling time N	discount coefficient	hidden layer size	learning rate (pre-training and joint-training)
HRRL	4	0.5	8	0.001/0.005
DARL	3	0.5	8	0.001/0.005

5.3.2. Ablation Experiment

Figures 6 and 7 illustrate the performance of HRL-based models concerning different top N values on two datasets. Within the category of HRL-based models, the ACHRL, ACHRL_H, and ACHRL_L models demonstrated superior performance in HR on two datasets. This confirms the effectiveness of our proposed model.

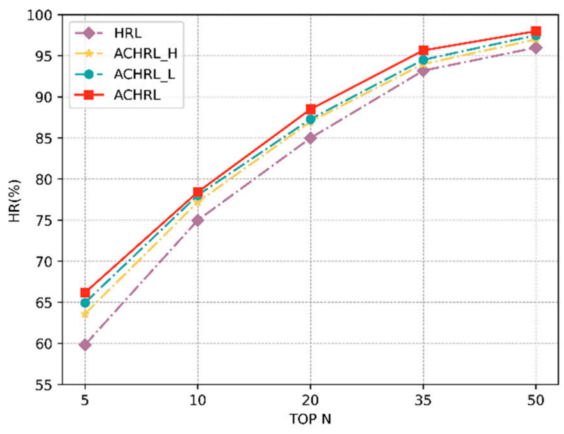


Figure 6. The performance of HRL-based models, assessed in terms of HR (%) at different top N HR values, on MOOCCourse.

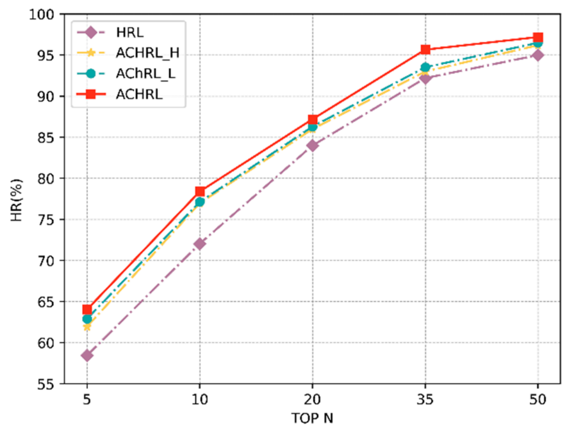


Figure 7. The performance of HRL-based models, assessed in terms of HR (%) at different top N HR values, on MOOCCube.

The ACHRL_H model outperforms the HRL model. This was because the high-level task modified the user profile more accurately after being optimized by the AC method. The ACHRL_L model also outperforms the HRL model because the noise courses in the course sequence are removed more accurately in the lower-level task. To sum up, the AC method optimizes two layers of tasks well.

As can be seen from the two figures, the ACHRL model attains the highest level of performance. This superiority is observed when compared to the other two variations of the method. Clearly, the combination of ACHRL_H and ACHRL_L maximize the accuracy of decision making for high-level and low-level tasks. The ACHRL model, which optimizes the two layers of tasks of the model, improves the accuracy of the decision task and achieves a more accurate course recommendation task.

Figures 8 and 9 show the variation in reward values of the ACHRL, ACHRL_H, and ACHRL_L models during the course of the experiment. It can be seen from the change in reward value in the figures that the ACHRL model performs best, which further confirms the effectiveness of the model for course recommendation.

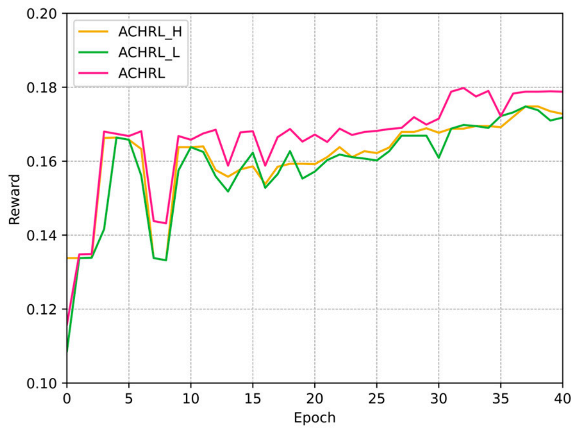


Figure 8. Variation in reward value with epoch of ACHRL, ACHRL_H, and ACHRL_L models on MOOCCourse.

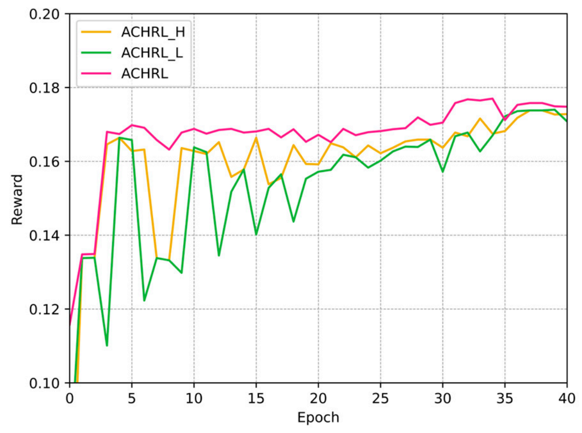


Figure 9. Variation in reward value with epoch of ACHRL, ACHRL_H, and ACHRL_L models on MOOCCube.

5.3.3. Influence of Hyper-Parameters

We examine the impact of two crucial hyperparameters (the size of the hidden layer of the attention mechanism network and the course embedding layer) on the model performance.

Figures 10 and 11 show the performance of the five models at various embedding layer sizes. In practice, the embedding layer size is specified as 8, 16, 32, 64, and 128. First of all, it can be seen from the two figures that the HRL-based model notably outperforms the NAIS model in HR. The ACHRL model achieves the best results on two datasets. The

ACHRL model handles historical courses more accurately and improves recommendation performance. In addition, our conclusion is that the recommendation performance of the five models in the experiment improves as the embedding layer size increases. This is because, with an increase in the dimension of the embedding layer, the attention mechanism's capacity for representation is enhanced, and the model can provide more useful information for recommendation learning.

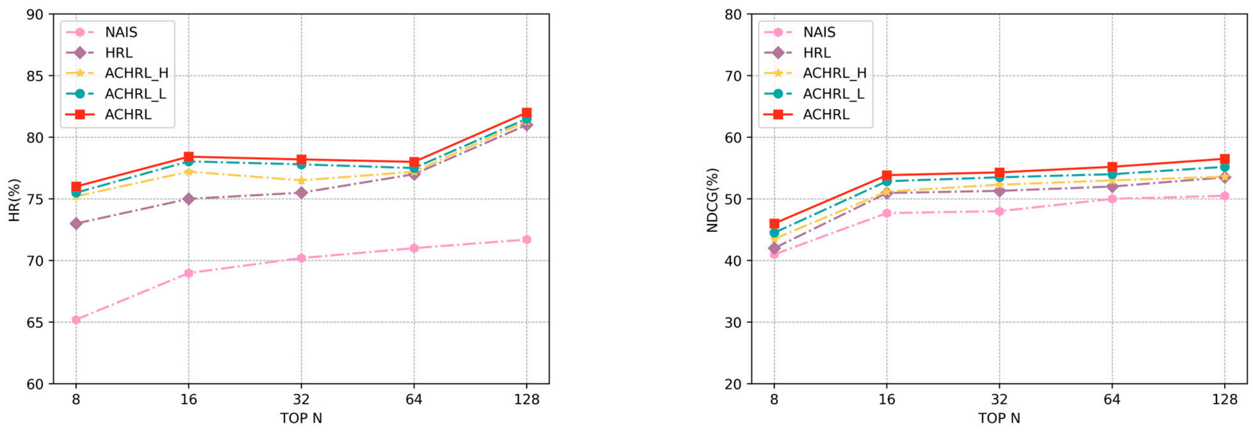


Figure 10. The performances of ACHRL, ACHRL_H, ACHRL_L, HRL, and NAIS, concerning different embedding sizes 8/16/32/64/128, are evaluated with respect to HR@10 and NDCG@10 on MOOCCourse.

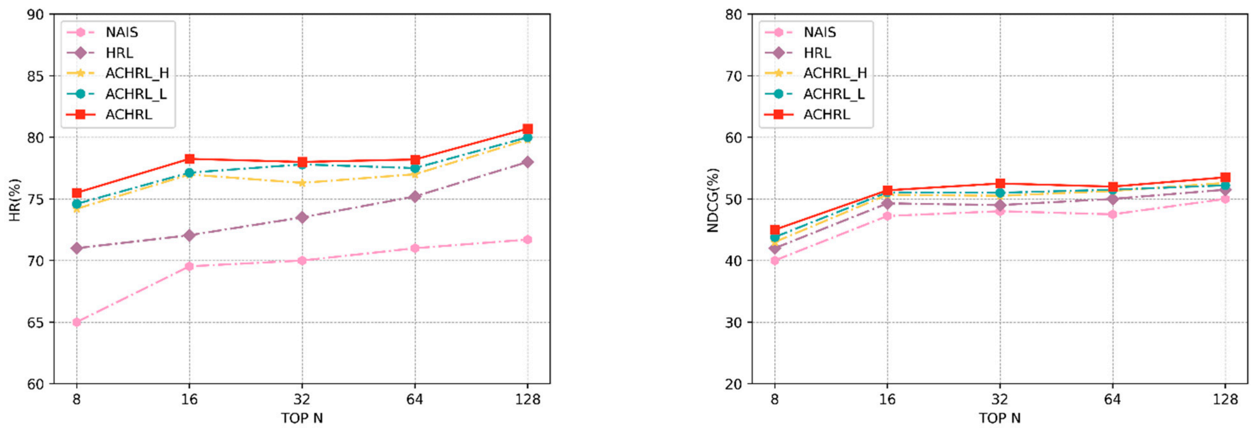


Figure 11. The performances of ACHRL, ACHRL_H, ACHRL_L, HRL, and NAIS, concerning different embedding sizes 8/16/32/64/128, are evaluated with respect to HR@10 and NDCG@10 on MOOCCube.

Figures 12 and 13 display the performance of the five models across various hidden layer sizes. In our experiment, we empirically chose 4, 8, 16, 32, and 64 as the hidden layer sizes, respectively. As shown in the two figures, the ACHRL model performed best relative to the other models on two datasets. It can be seen from the figures that our model is robust.

5.3.4. Performance Analysis

To visually illustrate the effectiveness of the HRL model, we provide specific instances of course recommendations for qualitative analysis. Table 4 displays the performances of the ACHRL and HRL models. ($\surd$ and $\times$ in the table indicate whether the recommended result is true or false) In the first case, the ACHRL model removes the noise course "Economics" accurately and recommends the target course correctly. In the second case, the ACHRL model removes the noise course "Operating Systems" and recommends the target course correctly while the HRL model did not accurately remove the noise course.

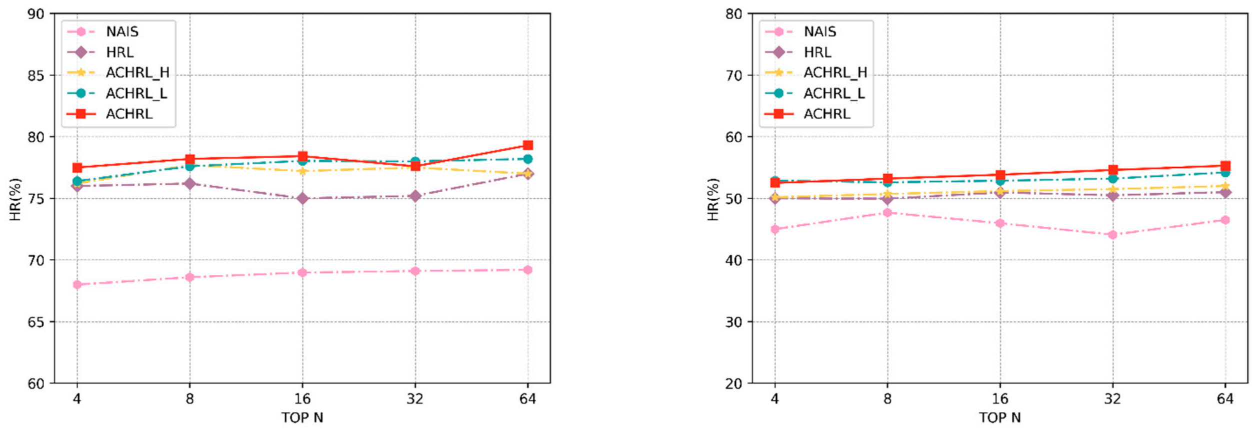


Figure 12. The performances of ACHRL, ACHRL_H, ACHRL_L, HRL, and NAIS, concerning different hidden layer sizes (4/8/16/32/64), are evaluated with respect to HR@10 and NDCG@10 on MOOCCourse.

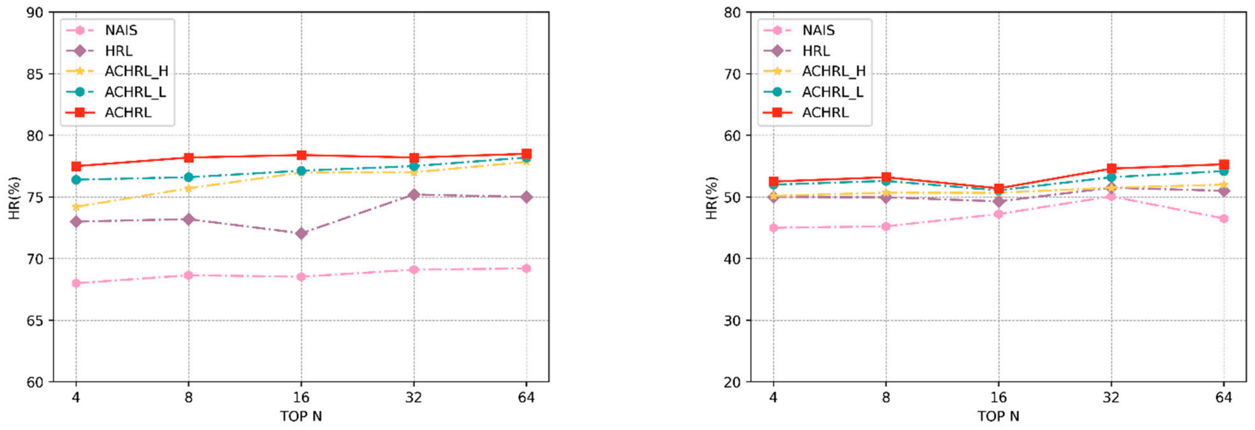


Figure 13. The performances of ACHRL, ACHRL_H, ACHRL_L, HRL, and NAIS, concerning different hidden layer sizes (4/8/16/32/64), are evaluated with respect to HR@10 and NDCG@10 on MOOCCube.

Table 4. Two cases of the recommendation performance of ACHRL and HRL.

	Model	Performance	Recommended Result
(1)	ACHRL	Data Structure, Java, Assembly Language, Software Engineering	Software Engineering (√)
	HRL	Data Structure, Java, Economics, Data Structure, Software Engineering	Organic Chemistry (×)
(2)	ACHRL	Monetary and Financial Studies, Investment Studies, Corporate Finance	Principles of Economics (√)
	HRL	Operating Systems, Monetary and Financial Studies, Investment Studies	Software Engineering (×)

This indicates the ACHRL model's capacity to effectively filter out irrelevant courses and offer recommendations that align better with the user's preferences.

6. Conclusions

In this paper, we introduce the ACHRL model for course recommendation. We are the first to apply the AC method to the hierarchical reinforcement learning model, reconstructing the user profile effectively. We applied the AC method to hierarchical tasks of the profile reviser, improving the accuracy of action selection at each layer, respectively. In addition, the use of the policy gradient method, which relies on the temporal difference error, leads to an enhancement in the recommendation performance. This gradient method can be applied not just for episodic scenarios but also in continuing situations. This allows the model to be used in more scenarios. The proposed ACHRL is compared with three latest state-of-the-art models, such as HRL, HRRL, and DARL, on two open datasets, and experimental results on the metrics validate the advantages of ACHRL. The model has good expansibility. For example, it can be applied to music, film, radio, and other fields. The historical data formed by the historical information of user interactions can be modelled well using the model method in this paper. Through the construction of a user profile, users can be recommended their favorite items.

We identify the limitation of our work. The metric used in this model to evaluate user satisfaction is relatively simple. While the ACHRL model aligns with the primary objectives of many existing recommendation models, which prioritize recommendation accuracy, the field of recommendation embraces a multitude of user satisfaction metrics. Users often appreciate diverse offerings, for example, ranging from popular courses at different universities to those from various majors. Consequently, a recommendation model that strikes a balance between precision and variety can significantly enhance user satisfaction.

Potential future work may investigate the following aspects. First, we plan to explore the ACHRL model's performance across multiple evaluation metrics, particularly focusing on recommendation accuracy and diversity. To achieve this, we intend to employ a multi-objective evolutionary approach for optimizing the evaluation process [58]. Moreover, in terms of methods, we have learned that the Advantage Actor-Critic (A2C) method [59] is an improvement of the Actor-Critic method designed to improve training efficiency and stability, and we will consider embedding the A2C method into our model to further improve the accuracy of model recommendations.

Author Contributions: Conceptualization, K.L.; Methodology and writing, G.Z.; Formal analysis, J.G.; Data curation, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (No. 62377036) and Science and Technology Program of Tianjin (No. A231005701).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Saadatdoost, R.; Sim, A.T.H.; Jafarkarimi, H.; Mei Hee, J. Exploring MOOC from education and Information Systems perspectives: A short literature review. *Educ. Rev.* **2015**, *67*, 505–518. [CrossRef]
2. Cheng, J.; Yuen, A.H.; Chiu, D.K. Systematic review of MOOC research in mainland China. *Libr. Hi Tech* **2022**, *41*, 1476–1497. [CrossRef]
3. Atiaja, L.A.; Proenza, R. The MOOCs: Origin, characterization, principal problems and challenges in Higher Education. *J. e-Learn. Knowl. Soc.* **2016**, *12*, 65–76.
4. Laurillard, D. The educational problem that MOOCs could solve: Professional development for teachers of disadvantaged students. *Res. Learn. Technol.* **2016**, *24*, 30–53. [CrossRef]
5. Xu, M.; Deng, J.; Zhao, T. On Status Quo, Problems, and Future Development of Translation and Interpreting MOOCs in China--A Mixed Methods Approach. *J. Interact. Media Educ.* **2020**, *2020*, 367–371. [CrossRef]
6. Parameswaran, A.; Venetis, P.; Garcia-Molina, H. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Trans. Inf. Syst. (TOIS)* **2011**, *29*, 1–33. [CrossRef]
7. Zhang, H.; Huang, T.; Lv, Z.; Liu, S.; Zhou, Z. MCRS: A course recommendation system for MOOCs. *Multimed. Tools Appl.* **2018**, *77*, 7051–7069. [CrossRef]
8. Jiang, W.; Pardos, Z.A.; Wei, Q. Goal-based course recommendation. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge, Tempe, AZ, USA, 4–8 March 2019; pp. 36–45.
9. Ma, B.; Lu, M.; Taniguchi, Y.; Konomi, S.I. CourseQ: The impact of visual and interactive course recommendation in university environments. *Res. Pract. Technol. Enhanc. Learn.* **2021**, *16*, 18. [CrossRef]

10. Thanh-Nhan, H.L.; Nguyen, H.H.; Thai-Nghe, N. Methods for building course recommendation systems. In Proceedings of the 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE), Hanoi, Vietnam, 6–8 October 2016; IEEE: Piscataway, NJ, USA; pp. 163–168.
11. Khalid, A.; Lundqvist, K.; Yates, A. A literature review of implemented recommendation techniques used in Massive Open online Courses. *Expert Syst. Appl.* **2022**, *187*, 115926. [CrossRef]
12. Zhang, J.; Hao, B.; Chen, B.; Li, C.; Chen, H.; Sun, J. Hierarchical reinforcement learning for course recommendation in MOOCs. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 435–442.
13. Kabbur, S.; Ning, X.; Karypis, G. Fism: Factored item similarity models for top-n recommender systems. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 659–667.
14. He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.G.; Chua, T.S. Nais: Neural attentive item similarity model for recommendation. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 2354–2366. [CrossRef]
15. Zhang, L.; Zhang, L. Top-N recommendation algorithm integrated neural network. *Neural Comput. Appl.* **2021**, *33*, 3881–3889. [CrossRef]
16. Zhao, X.; Zhang, Z.; Bi, X.; Sun, Y. A new point-of-interest group recommendation method in location-based social networks. *Neural Comput. Appl.* **2020**, *35*, 12945–12956. [CrossRef]
17. Jiang, X.; Sun, H.; Zhang, B.; He, L.; Jia, X. A novel meta-graph-based attention model for event recommendation. *Neural Comput. Appl.* **2022**, *34*, 14659–14682. [CrossRef]
18. Liu, H.; Wang, Y.; Lin, H.; Xu, B.; Zhao, N. Mitigating sensitive data exposure with adversarial learning for fairness recommendation systems. *Neural Comput. Appl.* **2022**, *34*, 18097–18111. [CrossRef]
19. Ren, Y.; Liang, K.; Shang, Y.; Zhang, Y. MulOER-SAN: 2-layer multi-objective framework for exercise recommendation with self-attention networks. *Knowl. Based Syst.* **2023**, *260*, 110117. [CrossRef]
20. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
21. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1057–1063.
22. O'Doherty, J.P.; Dayan, P.; Friston, K.; Critchley, H.; Dolan, R.J. Temporal difference models and reward-related learning in the human brain. *Neuron* **2003**, *38*, 329–337. [CrossRef]
23. Li, J.; Ye, Z. Course recommendations in online education based on collaborative filtering recommendation algorithm. *Complexity* **2020**, *2020*, 6619249. [CrossRef]
24. Ghauth, K.I.; Abdullah, N.A. The effect of incorporating good learners' ratings in e-Learning content-based recommender System. *J. Educ. Technol. Soc.* **2011**, *14*, 248–257.
25. Xu, G.; Jia, G.; Shi, L.; Zhang, Z. Personalized course recommendation system fusing with knowledge graph and collaborative filtering. *Comput. Intell. Neurosci.* **2021**, *2021*, 9590502. [CrossRef]
26. Emon, M.I.; Shahiduzzaman, M.; Rakib MR, H.; Shathee MS, A.; Saha, S.; Kamran, M.N.; Fahim, J.H. (2021, August). In Profile Based Course Recommendation System Using Association Rule Mining and Collaborative Filtering. In Proceedings of the 2021 International Conference on Science & Contemporary Technologies (ICSCT), Dhaka, Bangladesh, 5–7 August 2021; IEEE: Piscataway, NJ, USA; pp. 1–5.
27. Gao, M.; Luo, Y.; Hu, X. Online course recommendation using deep convolutional neural network with negative sequence mining. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 9054149. [CrossRef]
28. Wang, X.; Ma, W.; Guo, L.; Jiang, H.; Liu, F.; Xu, C. Hgnn: Hyperedge-based graph neural network for mooc course recommendation. *Inf. Process. Manag.* **2022**, *59*, 102938. [CrossRef]
29. Ren, X.; Yang, W.; Jiang, X.; Jin, G.; Yu, Y. A deep learning framework for multimodal course recommendation based on LSTM+ attention. *Sustainability* **2022**, *14*, 2907. [CrossRef]
30. Moerland, T.M.; Broekens, J.; Plaat, A.; Jonker, C.M. Model-based reinforcement learning: A survey. *Found. Trends® Mach. Learn.* **2023**, *16*, 1–118. [CrossRef]
31. Rohde, D.; Bonner, S.; Dunlop, T.; Vasile, F.; Karatzoglou, A. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv* **2018**, arXiv:1808.00720.
32. Wang, X.; Chen, W.; Wu, J.; Wang, Y.F.; Wang, W.Y. Video captioning via hierarchical reinforcement learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4213–4222.
33. Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4193–4206.
34. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Found. Trends® Mach. Learn.* **2018**, *11*, 219–354. [CrossRef]
35. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. In Proceedings of the AAAI Conference on Artificial Intelligence 2018, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
36. Li, S.E. Deep Reinforcement Learning. In *Reinforcement Learning for Sequential Decision and Optimal Control Singapore*; Springer Nature: Singapore, 2023; pp. 365–402.

37. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
38. Wiering, M.A.; Van Otterlo, M. Reinforcement learning. *Adapt. Learn. Optim.* **2012**, *12*, 729.
39. Mo, S.; Pei, X.; Wu, C. Safe reinforcement learning for autonomous vehicle using monte carlo tree search. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6766–6773. [CrossRef]
40. Wang, J.X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J.Z.; Munos, R.; Botvinick, M. Learning to reinforcement learn. *arXiv* **2016**, arXiv:1611.05763.
41. Pateria, S.; Subagdja, B.; Tan, A.H.; Quek, C. Hierarchical reinforcement learning: A comprehensive survey. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–35. [CrossRef]
42. Nachum, O.; Gu, S.S.; Lee, H.; Levine, S. Data-efficient hierarchical reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 3307–3317.
43. Botvinick, M.M. Hierarchical reinforcement learning and decision making. *Curr. Opin. Neurobiol.* **2012**, *22*, 956–962. [CrossRef]
44. Lin, Y.; Lin, F.; Zeng, W.; Xiahou, J.; Li, L.; Wu, P.; Miao, C. Hierarchical reinforcement learning with dynamic recurrent mechanism for course recommendation. *Knowl. Based Syst.* **2022**, *244*, 108546. [CrossRef]
45. Lin, Y.; Lin, F.; Yang, L.; Zeng, W.; Liu, Y.; Wu, P. Context-aware reinforcement learning for course recommendation. *Appl. Soft Comput.* **2022**, *125*, 109189. [CrossRef]
46. Lin, Y.; Feng, S.; Lin, F.; Zeng, W.; Liu, Y.; Wu, P. Adaptive course recommendation in MOOCs. *Knowl. Based Syst.* **2021**, *224*, 107085. [CrossRef]
47. Nachum, O.; Norouzi, M.; Xu, K.; Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 2772–2782.
48. Howard, R.A. *Dynamic Programming and Markov Processes*; John Wiley: Hoboken, NJ, USA, 1960.
49. Garcia, F.; Rachelson, E. Markov decision processes. In *Markov Decision Processes in Artificial Intelligence*; Wiley Online Library: Hoboken, NJ, USA, 2013; pp. 1–38.
50. Frome, A.; Corrado, G.S.; Shlens, J.; Bengio, S.; Dean, J.; Ranzato, M.A.; Mikolov, T. Devise: A deep visual-semantic embedding model. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2121–2129.
51. Lahitani, A.R.; Permanasari, A.E.; Setiawan, N.A. Cosine similarity to determine similarity measure: Study case in online essay assessment. In Proceedings of the 2016 4th International Conference on Cyber and IT Service Management, Bandung, Indonesia, 26–27 April 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–6.
52. Taud, H.; Mas, J.F. Multilayer perceptron (MLP). In *Geomatic Approaches for Modeling Land Change Scenarios*; Springer: Cham, Switzerland, 2018; pp. 451–455.
53. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
54. Poth, C.; Pfeiffer, J.; Rücklé, A.; Gurevych, I. What to pre-train on efficient intermediate task selection. *arXiv* **2021**, arXiv:2104.08247.
55. Liu, H.; Yu, J.; Chen, X.; Zhang, L. NeuMF: Predicting Anti-cancer Drug Response Through a Neural Matrix Factorization Model. *Curr. Bioinform.* **2022**, *17*, 835–847. [CrossRef]
56. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1419–1428.
57. Dalianis, H. Evaluation metrics and evaluation. In *Clinical Text Mining: Secondary Use of Electronic Patient Records*; Springer: Cham, Switzerland, 2018; pp. 45–53.
58. Von Lücken, C.; Barán, B.; Brizuela, C. A survey on multi-objective evolutionary algorithms for many-objective problems. *Comput. Optim. Appl.* **2014**, *58*, 707–756. [CrossRef]
59. Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M.; Soares, V.N.; Caldeira, J.M. Comparison of on-policy deep reinforcement learning A2C with off-policy DQN in irrigation optimization: A case study at a site in Portugal. *Computers* **2022**, *11*, 104. [CrossRef]