Hierarchical Reinforcement Learning for Course Recommendation in MOOCs

Jing Zhang[†⋆], Bowen Hao[†⋆], Bo Chen[†⋆], Cuiping Li[†⋆], Hong Chen[†⋆], Jimeng Sun[♯]
[†] Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, Renmin University of China
[⋆]Information School, Renmin University of China
[♯]Computational Science and Engineering at College of Computing, Georgia Institute of Technology
{zhang-jing, jeremyhao, bochen, licuiping, chong}@ruc.edu.cn, jsun@cc.gatech.edu

Abstract

The proliferation of massive open online courses (MOOCs) demands an effective way of personalized course recommendation. The recent attention-based recommendation models can distinguish the effects of different historical courses when recommending different target courses. However, when a user has interests in many different courses, the attention mechanism will perform poorly as the effects of the contributing courses are diluted by diverse historical courses. To address such a challenge, we propose a hierarchical reinforcement learning algorithm to revise the user profiles and tune the course recommendation model on the revised profiles.

Systematically, we evaluate the proposed model on a real dataset consisting of 1,302 courses, 82,535 users and 458,454 user enrolled behaviors, which were collected from XuetangX—one of the largest MOOCs in China. Experimental results show that the proposed model significantly outperforms the state-of-the-art recommendation models (improving 5.02% to 18.95% in terms of HR@10).

Introduction

Nowadays, massive open online courses, or MOOCs, are attracting widespread interest as an alternative education model. Lots of MOOCs platforms such as Coursera, edX and Udacity have been built and provide low cost opportunities for anyone to access a massive number of courses from the worldwide top universities. The proliferation of heterogeneous courses in MOOCs platforms demands an effective way of personalized course recommendation for their users.

The problem can be simply formalized as given a set of historical courses that were enrolled by a user before time t, we aim at recommending the most relevant courses that will be enrolled by the user at time $t+1$. We can view the historical enrolled courses as a user's profile, and the key factor of recommendation is to accurately characterize and model the user's preference from her profile. Many state-of-the-art algorithms have been proposed to model users' preferences in different ways. For example, when ignoring the order of the historical courses, we can adopt the factored item similarity model (FISM) (Kabbur, Ning, and Karypis 2013) to represent each course as an embedding vector and average the embeddings of all the historical courses to represent a user's preference. To capture the order of the courses, we can input a temporal sequence of the historical courses into the gated recurrent unit (GRU) model (Hidasi et al. 2016) and output the last embedding vector as the user preference. However, the model fidelity is limited by the assumption that all the historical courses play the same role at estimating the similarity between the user profile and the target course. To distinguish the effects of different courses, attention-based models such as neural attentive item similarity (NAIS) (He et al. 2018) and neural attentive session-based recommendation (NASR) (Li et al. 2017) can be used to estimate an attention coefficient for each historical course as its importance in recommending the target course.

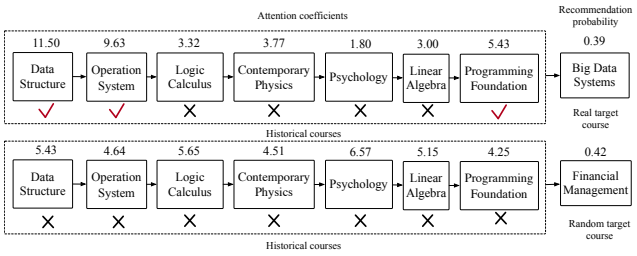


Figure 1: A motivating example of course recommendation. The scores on top of the historical courses are the attention coefficients calculated by NAIS and the scores on top of the target courses are the recommendation probabilities predicted by NAIS (He et al. 2018). The goal of this paper is to remove the courses with few contributions in a prediction as much as possible.

Although existing attention-based models improve the recommendation performance, it still poses unsolved challenges. Firstly, when a user enrolled diverse courses, the effects of the courses that indeed reflect the user's interest in the target course will be diluted by many irrelevant courses. For example, Figure 1 illustrates a recommendation result calculated by NAIS (He et al. 2018). The score on top of each historical course represents the calculated attention coefficient[1]. The real target course "Big Data Systems" is not successfully recommended in the top 10 ranked courses. Although the major contributing historical courses

[1]The sum of the attentions is normalized larger than 1 to lessen the punishment of active users (Cf. (He et al. 2018) for details).

like "Data Structure", "Operation System" and "Programming Foundation" are assigned relatively high attention coefficients, their effects are discounted by many other categories of courses such as psychology, physics and mathematics after aggregating all the historical courses by their attentions. Secondly, even if no historical courses can contribute in predicting a random target course, each historical course will still be rigidly assigned an attention coefficient, which may cause the random target course ranked before the real target one, as demonstrated by the random course "Financial Management" in Figure 1. In summary, the historical noisy courses that make small or even no contributions may disturb the prediction results significantly, even if they are assigned small attention coefficients.

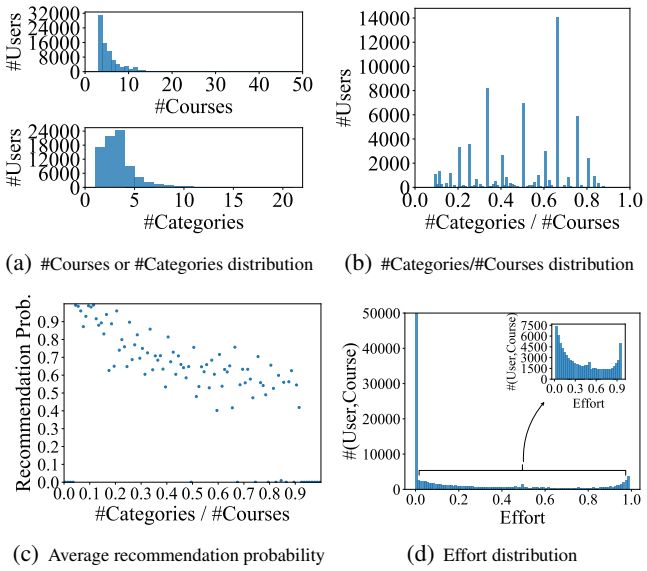
To deal with the above issues, we propose to revise user profiles by removing the noisy courses instead of assigning an attention coefficient to each of them. The key challenge is that we do not have explicit/supervised information about which courses from the history are noises and should be removed. We propose a hierarchal reinforcement learning algorithm to solve it. Specifically, we formalize the revising of a user profile to be a hierarchical sequential decision process. A high-level task and a low-level task are performed to remove the noisy courses, under the supervision of the feedback from the environment that consists of the dataset and a pre-trained basic recommendation model. Essentially, the profile reviser and the basic recommendation model are jointly trained together. Our contributions include:

- We propose a novel model for course recommendation in MOOCs, which consists of a profile reviser and a basic recommendation model. With joint training of the two models, we can effectively remove the noisy courses in user profiles.
- We propose a hierarchical reinforcement learning algorithm to revise the user profiles, which enables the model to remove the noise courses without explicit annotations.
- We collect a dataset, consisting of 1,302 courses, 82,535 users and 458,454 user enrolled behaviors, from XuetangX, one of the largest MOOCs in China, to evaluate the proposed model. Experimental results show that the proposed model significantly outperforms the state-of-the-art baselines (improving 5.02% to 18.95% in terms of HR@10).

MOOC Data

We collect the dataset from XuetangX[2], one of the largest MOOCs platforms in China. We unify the same courses offered in different years such as "Data Structure(2017)" and "Data Structure(2018)" into one course and only select the users who enrolled at least three courses from October 1st, 2016 to March 31st, 2018. The resulting dataset consists of 1,302 courses which belonging to 23 categories, 82,535 users and 458,454 user-course pairs. We also collect the duration of each video in a course watched by a user. Before training the model, we conduct a series of analyses to investigate why we need to revise the user profiles.

[2]http://www.xuetangx.com



(a) #Courses or #Categories distribution (b) #Categories/#Courses distribution

(c) Average recommendation probability (d) Effort distribution

Figure 2: Data distributions.

Figure 2(a) presents the distribution of the enrolled course number of a user in the top and the distribution of the category number of the enrolled courses in the bottom. Then we calculate the ratio between the category number and the course number as the category ratio of a profile to represent the attentiveness of a user. A bigger category ratio indicates the user is more distractive, while a smaller category ratio indicates the user is more attentive. Figure 2(b) shows the distribution of the category ratio. From the three figures, we can see that although a large number of users enrolled a small number of courses and categories, the ratio between them is relatively evenly distributed. We further average the probabilities of recommending a real target course calculated by NAIS (He et al. 2018) for the user profiles of the same category ratio and present the probability distribution over category ratio in Figure 2(c). We can see that the probability decreases with the increase of the category ratio. In summary, all these analyses indicate that a large number of users enrolled diverse courses, and the recommendation performance based on these diverse profiles is impacted. Thus, we have to study how to revise the user profiles. In addition, we calculate the ratio between the watch duration and the total duration of a video as the watch ratio, and use the maximal watch ratio of all the videos in a course to represent the effort taken by the user in the course. We present the effort distribution of user enrolled courses in Figure 2(d) and the filtered effort distribution (i.e., effort larger than 0.01) in the embeded subfigure, which indicate that users take distinguished effort in different courses. The phenomenon can guide the design of the agent policy later.

Background: Recommendation Models

Problem Formulation

Let $U = \{u_1, \cdots, u_{|U|}\}$ be a set of users and $C = \{c_1, \cdots, c_{|C|}\}$ be a set of courses in the MOOCs platform.

For each user u, given her profile, i.e., the historical enrolled courses $\mathcal{E}^u := (e_1^u, \cdots, e_{t_u}^u)$ with $e_t^u \in C$, we are aiming at recommending the courses u would enroll at next time $t_u + 1$. We deal with the relative time instead of the absolute time the same as (Rendle, Freudenthaler, and Schmidt-Thieme 2010).

The Basic Recommendation Model

The key factor of recommendation is to accurately characterize a user's preference according to her profile $\mathcal{E}^u$. The general idea is, we represent each historical course e_t^u as a real-valued low dimensional embedding vector $\mathbf{p}_t^u$, and aggregate the embeddings of all the historical courses $\mathbf{p}_1^u, \ldots, \mathbf{p}_t^u$ to represent user u's preference $\mathbf{q}_u$. If we also represent a target course c_i as an embedding vector $\mathbf{p}_i$, the probability of recommending course c_i to user u, i.e.,$P(y = 1|\mathcal{E}^u, c_i)$, can be calculated as:

$$P(y = 1|\mathcal{E}^u, c_i) = \sigma(\mathbf{q}_u^T \mathbf{p}_i), \tag{1}$$

where $y = 1$ indicates that c_i is recommended to user u and σ is the sigmoid function to transform the input into a probability. Then the key issue is how to obtain the aggregated embedding $\mathbf{q}_u$. One straightforward way is to average the embeddings of all the historical courses, i.e. $\mathbf{q}_u = \frac{1}{t_u} \sum_{t=1}^{t_u} \mathbf{p}_t^u$. However, equally treating all the courses' contributions may impact the representation of a user's real interest in a target course. Thus, as NAIS (He et al. 2018) does, we can adopt the attention mechanism to estimate an attention coefficient a_{it}^u for each historical course e_t^u when recommending c_i. Specifically, we parameterize the attention coefficient a_{it}^u as a function with $\mathbf{p}_t^u$ and $\mathbf{p}_i$ as inputs and then aggregate the embeddings according to their attentions:

$$\mathbf{q}_u = \sum_{t=1}^{t_u} a_{it}^u \mathbf{p}_t^u, \; a_{it}^u = f(\mathbf{p}_t^u, \mathbf{p}_i), \tag{2}$$

where f can be instantiated by a multi-layer perception on the concatenation or the element-wise product of the two embeddings $\mathbf{p}_t^u$ and $\mathbf{p}_i$.

We can also adopt NASR (Li et al. 2017)—an attentive recurrent neural networks to capture the order of the historical courses. Specifically, at each time t, NASR outputs a hidden vector $\mathbf{h}_t^u$ to represent a user's preference until time t based on both the course enrolled at time t and all the previous courses before t. Then the same attention mechanism is applied on the hidden vectors of all the timestamps.

The Proposed Model

In this section, we firstly give an overview of the proposed model, then we introduce a hierarchal reinforcement learning algorithm to revise user profiles, and finally explain the training process of the entire model.

Overview

Although the basic recommendation models can estimate an attention coefficient for each historical course, the effects of the contributing courses to the target one may be diluted by the irrelevant ones when users enrolled many diverse courses. To deal with this issue, we propose a model to revise the user profiles by removing the noisy courses from the history, and recommend courses based on the revised profiles. The key challenge is how to determine which historical courses are the noises without direct supervision, i.e., identify the courses that disturb the recommendation performance. Thus, we propose a hierarchical reinforcement learning algorithm to solve it. Specifically, we formalize the revising process of a user profile to be a hierarchical sequential decision process by an agent. Following a revising policy, a high-level and a low-level task are performed to revise the profile. After the whole profile of a user is revised, the agent gets a delayed reward from the environment, based on which it updates its policy. The environment can be viewed as the dataset and a pre-trained basic recommendation model as introduced in the previous section. After the policy is updated, the basic recommendation model is re-trained based on the profiles revised by the agent. Essentially, the profile reviser and the recommendation model are jointly trained. Figure 3 illustrates the framework of the proposed model.

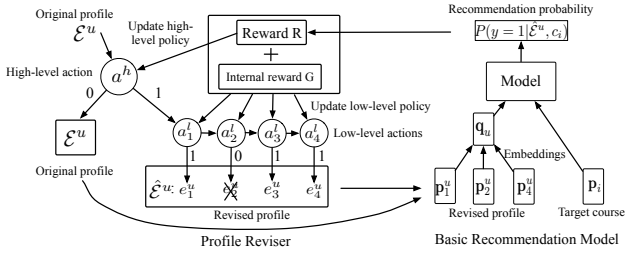


Figure 3: The overall framework of the proposed model.

Profile Reviser

As mentioned before, the profile reviser aims to remove the noisy courses with few contributions in a prediction. Inspired by the theory of hierarchical abstract machines (Parr and Russell 1998), we cast the task of profile reviser as a hierarchical Markov Decision Process (MDP). Generally speaking, we decompose the overall task MDP M into two kinds of subtasks M^h and M^l, where M^h is the high-level abstract task in the hierarchy and solving it solves the entire MDP M, and M^l is the low-level primitive task in the hierarchy. Each kind of task is defined as a 4-tuple MDP ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$), where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T}$ is a transition model mapping $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ into probabilities in [0,1], and $\mathcal{R}$ is a reward function mapping $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ into real-valued rewards. We formulate our task by a high-level task and a low-level task. Specifically, given a sequence of historical courses $\mathcal{E}^u := (e_1^u, \cdots, e_{t_u}^u)$ of user u and the target course c_i, the agent performs a high-level task of one binary action to determine whether to revise the whole profile $\mathcal{E}^u$ or not. If it decides to revise $\mathcal{E}^u$, the agent performs a low-level task of multiple actions to determine whether to remove each historical course $e_t^u \in \mathcal{E}^u$ or not. After the low-level task is finished, the overall task is finished. If the high-level task decides to make no revision, low-level task

will not be executed and the overall task is directly finished.

We formulate the profile reviser as two-level MDPs, because a part of the user profiles are discriminative and can be already correctly predicted by the basic recommendation model. As presented in Figure 2(b), about 30% users are relatively attentive (i.e., #Categories/#Courses<0.4) and the corresponding average probabilities of recommending the real target courses are high (i.e., larger than 0.6). We can simply keep those profiles as the original ones and only revise the indiscriminative ones. Out of this consideration, we design a high-level task to decide whether to revise the whole profile of a user or not, and a low-level task to decide which course in the profile should be removed.

Note that for both the high-level and low-level task, given a state and an action, it will transit to a determined state with probability 1. We will introduce the details of how to design the state, action and reward for the two-level tasks.

State. The high-level task takes an action according to the state of the whole profile $\mathcal{E}^u$ and the low-level task takes a sequence of actions according to the state of each course $e_t^u \in \mathcal{E}^u$. We define different state features for the two tasks.

- **Low-level task:** When determining to remove a historical course $e_t^u \in \mathcal{E}^u$, we define the state features $\mathbf{s}_t^l$ as the cosine similarity between the embedding vectors of the current historical course e_t^u and the target course c_i, the element-wise product between them, and also the average of the two previous features over all the reserved historical courses, where the embedding vector of a course $\mathbf{p}_i$ can be provided by a pre-trained basic recommendation model. We also define the effort taken in a course as an additional state feature, as the effort taken in a course is significantly different (Cf. Figure 2(d)) and it may also indicate the contribution of e_t^u to c_i besides the similarity-based features. For simplicity and clarity, we ignore the superscript u in all the notations about the state features.
- **High-level task:** When determining to revise a whole profile $\mathcal{E}^u$, we define the state features $\mathbf{s}^h$ as the average cosine similarity between the embedding vectors of each historical course in $\mathcal{E}^u$ and the target course and the average element-wise product between them. We also define an additional state feature as the probability $P(y = 1|\mathcal{E}^u, c_i)$ of recommending c_i to user u by a basic recommendation model. The probability reflects how credible the course c_i will be recommended based on the profile $\mathcal{E}^u$. The lower recommendation probability is, more effort should be taken to revise $\mathcal{E}^u$. Note we train the profile reviser only based on the positive instances, i.e., a user profile paired with a real target course, as negative instances with random target courses can hardly guide the agent to select the contributing courses to the target course. Thus $P(y = 0|\mathcal{E}^u, c_i)$ for a negative instance is not calculated.

Action and Policy. We define the high-level action $a^h \in \{0, 1\}$ as a binary value to represent whether to revise the whole profile of a user or not, and define a low-level action $a_t^l \in \{0, 1\}$ as a binary value to represent whether to remove the historical course e_t^u or not. We perform a low-level action a_t^l according to the policy function as follows:

$$\begin{aligned} \mathbf{H}_t^l &= \text{ReLU}(\mathbf{W}_1^l \mathbf{s}_t^l + \mathbf{b}^l), \\ \pi(\mathbf{s}_t^l, a_t^l) &= P(a_t^l|\mathbf{s}_t^l, \Theta^l) \\ &= a_t^l \sigma(\mathbf{W}_2^l \mathbf{H}_t^l) + (1 - a_t^l)(1 - \sigma(\mathbf{W}_2^l \mathbf{H}_t^l)), \end{aligned} \tag{3}$$

where $\mathbf{W}_1^l \in \mathbb{R}^{d_1^l \times d_2^l}$, $\mathbf{W}_2^l \in \mathbb{R}^{d_2^l \times 1}$ and $\mathbf{b}^l \in \mathbb{R}^{d_2^l}$ are the parameters to be learned with d_1^l as the number of the state features and d_2^l as the dimension of the hidden layer. Notation $\mathbf{H}_t^l$ represents the embedding of the input state. We denote $\Theta^l = \{\mathbf{W}_1^l, \mathbf{W}_2^l, \mathbf{b}^l\}$. Sigmoid function σ is used to transform the input into a probability. The high-level action is performed according to the similar policy function with different parameters $\Theta^h = \{\mathbf{W}_1^h, \mathbf{W}_2^h, \mathbf{b}^h\}$.

Reward. The reward is a signal to indicate whether the performed actions are reasonable or not. We assume that every low-level action in the low-level task has a delayed reward after the last action $a_{t_u}^l$ is performed for the last course $e_{t_u}^u \in \mathcal{E}^u$. In another word, the immediate reward for a low-level action is zero except the last low-level action. Thus, we define the reward for each low-level action as:

$$R(a_t^l, \mathbf{s}_t^l) = \begin{cases} \log p(\hat{\mathcal{E}}^u, c_i) - \log p(\mathcal{E}^u, c_i). & \text{if } t = t_u; \\ 0 & \text{otherwise,} \end{cases}$$

where $p(\mathcal{E}^u, c_i)$ is an abbreviation of $p(y = 1|\mathcal{E}^u, c_i)$ and $\hat{\mathcal{E}}^u$ is the revised profile, which is a subset of $\mathcal{E}^u$. For the special case $\hat{\mathcal{E}}^u = \varnothing$, i.e., all the historical courses are removed, we randomly select a course from the original set $\mathcal{E}^u$. The reward is defined as the difference between the log-likelihood after and before the profile is revised. A positive difference indicates a positive utility gained by the revised profile.

If the high-level task chooses the revising action, it calls the low-level task and receives the same delayed reward $R(a_t^l, \mathbf{s}_t^l)$ after the last low-level action is performed. Otherwise, it keeps the original profile and obtain a zero reward as $\log p(\mathcal{E}^u, c_i)$ is not changed.

In addition, we define an internal reward $G(a_t^l, \mathbf{s}_t^l)$ which is used only inside the low-level task to speed up its local learning and does not propagate to the high-level task (Ghavamzadeh and Mahadevan 2003). Specifically, we first calculate the average cosine similarity between each historical course and the target course after and before the profile is revised, and then use the difference between them as the internal reward $G(a_t^l, \mathbf{s}_t^l)$. The internal reward encourages the agent to select the most relevant courses to the target course. Finally, we sum $G(a_t^l, \mathbf{s}_t^l)$ and $R(a_t^l, \mathbf{s}_t^l)$ as the reward for the low-level task.

Objective Function. We aim at finding the optimal parameters of the policy function defined in Eq. (3) to maximize the expected reward, i.e.,

$$\Theta^* = \text{argmax}_\Theta \sum_\tau P_\Theta(\tau; \Theta) R(\tau), \tag{4}$$

where Θ represents either Θ^h or Θ^l, τ is a sequence of the sampled actions and the transited states, $P_\Theta(\tau; \Theta)$ denotes the corresponding sampling probability and $R(\tau)$ is the reward for the sampled sequence τ. The sampled sequence τ can be $\{s_1^l, a_1^l, s_2^l, \cdots, s_t^l, a_t^l, s_{t+1}^l, \cdots\}$ for the low-level

Pre-train the basic recommendation model;
Pre-train the profiler reviser by running Algorithm 2 with the basic recommendation model fixed;
Jointly train the two models together by running Algorithm 2;

Algorithm 1: The Overall Training Process

Input: Training data $\{\mathcal{E}^1, \mathcal{E}^2, \cdots, \mathcal{E}^{|U|}\}$, a pre-trained basic recommendation model and a profile reviser parameterized by Φ^0 and Θ^0 respectively
Initialize $\Theta = \Theta^0$, $\Phi = \Phi^0$;
for *episode l=1 to L* **do**
  foreach $\mathcal{E}^u := (e_1^u, \cdots, e_{t_u}^u)$ *and* c_i **do**
    Sample a high-level action a^h with Θ^h;
    if $a^h = 0$ **then**
      $R(\mathbf{s}^h, a^h) = 0$
    else
      Sample a sequence of low-level actions $\{a_1^l, a_2^l, \cdots, a_{t_u}^l\}$ with Θ^l;
      Compute $R(a_{t_u}^l, \mathbf{s}_{t_u}^l)$ and $G(a_{t_u}^l, \mathbf{s}_{t_u}^l)$;
      Compute gradients by Eq. (5) and (6);
    end
  end
  Update Θ by the gradients;
  Update Φ in the basic recommendation model;
end

Algorithm 2: The Hierarchical Reinforcement Learning

task and $\{s^h, a^h\}$ for the high-level task. Since there are too many possible action-state trajectories for the entire sequences of the two tasks, we adopt the policy gradient theorem (Sutton et al. 2000) and the monto-carlo based policy gradient method (Williams 1992) to sample M action-state trajectories, based on which we calculate the gradient of the parameters for the low-level policy function:

$$\nabla_\Theta = \frac{1}{m}\sum_{m=1}^{M}\sum_{t=1}^{t_u}\nabla_\Theta \log \pi_\Theta(\mathbf{s}_t^m, a_t^m)(R(a_t^m, \mathbf{s}_t^m)+G(a_t^m, \mathbf{s}_t^m)), \tag{5}$$

where the reward $R(a_t^m, \mathbf{s}_t^m) + G(a_t^m, \mathbf{s}_t^m)$ for each action-state pair in sequence $\tau^{(m)}$ is assigned the same value and equals to the terminal reward $R(a_{t_u}^m, \mathbf{s}_{t_u}^m) + G(a_{t_u}^m, \mathbf{s}_{t_u}^m)$. The gradient for the high-level policy function:

$$\nabla_\Theta = \frac{1}{m}\sum_{m=1}^{M}\nabla_\Theta \log \pi_\Theta(\mathbf{s}^m, a^m)R(a_t^m, \mathbf{s}_t^m), \tag{6}$$

where the reward $R(a^m, \mathbf{s}^m)$ is assigned as $R(a_{t_u}^m, \mathbf{s}_{t_u}^m)$ when $a^m = 1$, and 0 otherwise. We omit the superscript ^{l} and ^{h} in Eq. (6) and (5) for simplicity.

Model Training

The two models of the profile reviser and the basic recommendation model are interleaved together, and we need to train them jointly. The training process is shown in Algorithm 1, where we firstly pre-train the basic recommendation model based on the original dataset, then we fix the parameters of the basic recommendation model and pre-train the profile reviser to automatically revise the user profiles; finally, we jointly train the models together. Same as the settings of (Feng et al. 2018), to have a stable update, each parameter is updated by a linear combination of its old version and the new old version, i.e., $\Theta_{new} = \lambda\Theta_{new} + (1-\lambda)\Theta_{old}$, where $\lambda \ll 1$. The time complexity is $O(L(N\bar{t}_u M))$, where L is the number of epochs, N is the number of instances, $\bar{t}_u$ is the the average number of historical courses and M is the Monto Carlo sampling time.

Experiments

Experimental Settings

Settings. The dataset is introduced in the section of MOOC data. We select the enrolled behaviors from October 1st, 2016 to December 30th, 2017 as the training set, and those from January 1st, 2018 to March 31st, 2018 as the test set. Each instance in the training or the test set is a sequence of historical enrolled courses paired with a target course. During the training process, for each sequence in the training data, we hold out the last course as the target course, and the rest are treated as the historical courses. For each positive instance, we construct 4 negative instances by replacing the target course with each of 4 randomly sampled courses. During the test process, we treat each enrolled course in the test set as the target course, and the corresponding courses of the same user in the training set as the historical courses. Each positive instance in the test set is paired with 99 randomly sampled negative instances (He et al. 2018).

Baseline Methods. The comparison methods include:

BPR (Rendle et al. 2009): optimizes a pairwise ranking loss for the recommendation task in a Bayesian way.

MLP (He et al. 2017): applies a multi-layer perceptron (MLP) on a pair of user and course embeddings to learn the probability of recommending the course to the user.

FM (Rendle 2012): is a principled approach that can easily incorporate any heuristic features. But for fair comparison, we only use the embeddings of users and courses.

FISM (Kabbur, Ning, and Karypis 2013): is an item-to-item collaborative filtering algorithm which conducts recommendation based on the average embedding of all the historical courses and the embedding of the target course.

NAIS (He et al. 2018): is also an item-to-item collaborative filtering algorithm but distinguishes the weights of different historical courses by an attention mechanism.

GRU (Hidasi et al. 2016): is a gated recurrent unit model that receives a sequence of historical courses as input and output the last hidden vector as the representation of a user's preference.

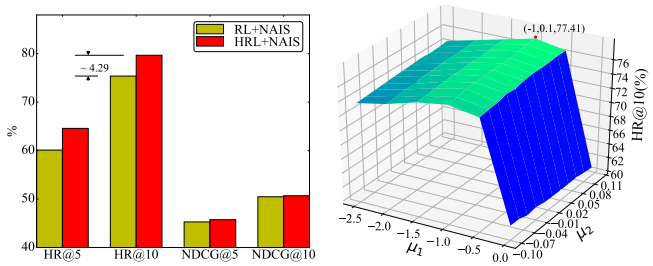
NASR (Li et al. 2017): is an improved GRU model that estimates an attention coefficient for each historical course based on the corresponding hidden vector output by GRU.

HRL+NAIS: is the proposed model that adopts NAIS as the basic recommendation model and we jointly train it with the hierarchical reinforcement learning (HRL) based profile reviser.

HRL+NASR: is also the proposed model but adopts NASR as the basic recommendation model.

Table 1: Recommendation performance (%).

Methods	HR@5	HR@10	NDCG@5	NDCG@10
BPR	46.82	60.73	34.16	38.65
MLP	52.16	66.29	40.39	44.41
FM	46.01	61.07	35.28	40.15
FISM	52.73	65.64	40.00	44.98
GRU	52.07	68.63	38.92	46.30
NAIS	56.42	69.05	43.73	47.82
NASR	54.64	69.48	42.39	47.33
HRL+NAIS	**64.59**	**79.68**	**45.74**	**50.69**
HRL+NASR	**59.05**	**74.50**	**47.51**	**52.73**



(a) RL+NAIS (b) Greedy+NAIS

Figure 4: Recommendation performance of model variants.

Evaluation Metrics. We evaluate all the methods in terms of the widely used metrics Hit Ratio of top K items (HR@K) and Normalized Discounted Cumulative Gain of top K items (NDCG@K), where HR@K is a recall-based metric that measures the percentage of the ground truth instances that are successfully recommended in top-K, and NDCG@K is a precision-based metrics that accounts for the predicted position of the ground truth instance (Huang et al. 2018; He et al. 2018; Rendle, Freudenthaler, and Schmidt-Thieme 2010). We set K as 5 and 10 and calculate all the metrics for every 100 instances (1 positive plus 99 negatives) and report the average score of all the users.

Implementaion Details. We implement the model by Tensorflow and run the code on an Enterprise Linux Server with 40 Intel(R) Xeon(R) CPU cores (E5-2630 and 512G memory) and 1 NVIDIA TITAN V GPU core (12G memory). For the profile reviser, sampling time M is set as 3, the learning rate is set as 0.001/0.0005 at the pre-training and joint-training stage respectively. In the policy function, the dimensions of the hidden layer d_2^l and d_2^h are both set as 8. For the basic recommender, the dimension of the course embeddings is set to 16, the learning rate is 0.01 at both the pre-training and joint-training stage, and the size of the mini-batch is 256. The delayed coefficient λ for the joint-training is 0.0005. The code is online now[3].

Performance Analysis

Overall Prediction Performance. Table 1 shows the overall performance of all the comparison methods. The proposed model performs clearly better than the comparison baselines (improving 5.02% to 18.95% in HR@10). The user-to-item based collaborative filtering methods such as BPR, MLP and FM perform the worst among all the methods, because in our dataset, most of the users only enrolled a few courses (i.e., less than 10 courses as shown in Figure 2(a)). Thus the embeddings for many users can not be sufficiently inferred from the sparse data. Among all the item-to-item based collaborative filtering methods, FISM and GRU perform worse than the others, as they make equal treatments on all the historical enrolled courses and thus the preference representation ability is limited. NAIS and NASR distinguish the effects of different historical courses by assigning them different attention coefficients. However, the useless courses will dilute the effects of the useful courses in the history when users enrolled many diverse courses. The proposed methods, HRL+NAIS and HRL+NASR perform the best, as they forcely remove the noisy courses instead of assigning soft attention coefficients, which distinguish the useful and useless courses significantly.

For the proposed methods, processing 1 episode of profile update requires 50-80 seconds and the recommender update requires 20-30 seconds. The best recommendation performance on test set is reached after about 20 episodes of recommender pre-training, 20 episodes of profile reviser pre-training and 5 episodes of joint training through the data, which totally requires 30-45 minutes of joint training.

Compared with One-level RL. We compare the proposed HRL with an one-level RL algorithm, which only uses the low-level task to directly decide to remove each course or not. The comparison results in Table 4(a) show that HRL outperforms the one-level RL. We find that for HRL, the average #Categories/#Courses of the revised profiles is 0.73 and that for the one-level RL is 0.75, which indicates that the revised profiles by the proposed HRL are more consistent (The larger the value is, the more diverse a profile is). This is because HRL uses an additional high-level task to decide to keep the consistent profiles and revise the diverse profiles. To verify whether the high-level task takes effect or not, we further check the difference between the kept profiles and the revised profiles decided by the high-level task. The average #Categories/#Courses of the kept profiles is 0.57, and that of the revised profiles is 0.69, which indicates that the high-level task tends to keep more consistent profiles while revise more diverse profiles.

Compared with Greedy Revision. We compare the proposed HRL with a greedy revision algorithm, which firstly decides to revise the whole profile $\mathcal{E}^u$ if $\log P(y = 1|\mathcal{E}^u, c_i) < \mu_1$, and further removes a $e_t^u \in \mathcal{E}^u$ if its cosine similarity with c_i is less than μ_2. In Figure 4(b), we tune μ_1 from -2.5 to 0 with interval 0.5, and tune μ_2 from -0.1 to 0.1 with interval 0.04, and obtain the best HR@10 as 77.44% when μ_1=-1 and μ_2=0.1, which is 2.27% less than HRL+NAIS. Note the best performance is obtained when the number of the remaining courses is almost the same as

[3]https://github.com/jerryhao66/HRL

Table 2: Case studies of the profiles revised by HRL+NAIS and the attention coefficients learned by NAIS.

Methods	Revised profile or the learned attentions	The target course
HRL+NAIS	~~Crisis Negotiation~~, ~~Social Civilization~~, Web Technology, C++ Program	Web Development
NAIS	Crisis Negotiation(29.61), Social Civilization(29.09), Web Technology(28.32), C++ Program(28.12)	Web Development
HRL+NAIS	Modern Biology, Medical Mystery, Biomedical Imaging, ~~R Program~~	Biology
NAIS	Modern Biology(37.79), Medical Mystery(37.96), Biomedical Imaging(37.62), R Program(37.84)	Biology
HRL+NAIS	~~Web Technology~~, Art Classics, ~~National Unity Theory~~, Philosophy	Life Aesthetics
NAIS	Web Technology(38.32), Art Classics(35.87), National Unity Theory(40.63), Philosophy(43.69)	Life Aesthetics

those by HRL+NAIS.

Compared with Attentions Coefficients. We present several cases of the revised profiles by the proposed HRL+NAIS and show three cases and the corresponding learned attention coefficients by NAIS in Table 2. The cases present that HRL+NAIS can definitely remove the noisy courses in the profile that are totally irrelevant to the target course. In contrary, although NAIS assigns high attentions to the contributing historical courses, the attentions of some other irrelevant courses are not significantly different, or even higher than the relevant ones, thus the effects of the real contributing courses are discounted after aggregating all the historical courses by their attentions. As a result, the performance of the recommendation model based on the discriminative revised profiles is improved.

Related Work

Collaborative filtering (CF) is widely used to do recommendation. User-to-item based CF, such as matrix factorization (Koren, Bell, and Volinsky 2009), bayesian personalized ranking (BPR) (Rendle et al. 2009) and factorization machine (FM) (Rendle 2012) performs recommendation based on both the user and item embeddings. These shallow models are further extended to deep neural network models (He et al. 2017; Guo et al. 2017; Zhang, Du, and Wang 2016). The user-to-item CF suffers from the sparsity of users' profiles. On the contrary, the item-to-item CF does not need to estimate user embeddings, and is heavily adopted in industrial applications (Davidson et al. 2010; Smith and Linden 2017). Early item-to-item based CF uses heuristic metrics such as Pearson coefficient or cosine similarity to estimate item similarities (Sarwar et al. 2001), followed by a machine learning method which calculates the item similarity as the dot product of item embeddings (Kabbur, Ning, and Karypis 2013). Sequential based models such as RNN (Tan, Xu, and Liu 2016) and GRU (Hidasi et al. 2016) are proposed to capture the temporal factor. Then the attention-based models such as NAIS (He et al. 2018) and NASR (Li et al. 2017) are further proposed to distinguish the effects of different items. Several researches are conducted on MOOCs platforms, such as learning behavior analysis (Anderson et al. 2014; Qiu et al. 2016; Qi et al. 2018) and course recommendation (Jing and Tang 2017). They focus on extracting features from multi-mode data sources besides the enrolled behaviors of users, thus it is unfair to compare with their methods.

Recently, some researchers attempt to adopt the reinforcement learning algorithm to solve many kinds of problems, such as relation classification (Feng et al. 2018), text classification (Zhang, Huang, and Zhao 2018), information extraction (Narasimhan, Yala, and Barzilay 2016), question answering (Wang et al. 2018b) and treatment recommendation (Wang et al. 2018a). Inspired by these successful attempts, we propose a hierarchal reinforcement learning algorithm to conduct course recommendation. Hierarchical reinforcement learning aims at decomposing complex tasks into multiple small tasks to reduce the complexity of decision making (Barto and Mahadevan 2003), where different HRLs such as option-based HRL that formulates the abstract knowledge and action as options (Sutton, Precup, and Singh 1999) and the hierarchical abstract machines (HAMs) that decomposes high-level activities into low-level activities (Sutton, Precup, and Singh 1999) are proposed. We formalize our problem by the theory of HAMs.

Conclusion

We present the first attempt to solve the problem of course recommendation in MOOCs platform by a hierarchical reinforcement learning model. The model jointly trains a profile reviser and a basic recommendation model, which enables the recommendation model being trained on user profiles revised by the profile reviser. With the designed two-level tasks, the agent in the hierarchical reinforcement learning model can effectively remove the noisy courses and reserve the real contributing courses to the target course.

We will try the proposed model in other domains. For example, people usually watch diverse movies, read diverse books and purchase diverse products. In those scenarios, we can imagine the need for selecting the most contributing historical items from users' diverse profiles, which poses the same challenges with the recommendations in MOOCs. In the future, we will also explore how to connect the courses in MOOCs to the external entities or knowledge such as the academic papers and researchers (Tang et al. 2008) to enable more accurate course recommendation in MOOCs.

Acknowledgments. We thank Xuetang.com for sharing the datasets. This work is supported by National Key R&D Program of China (No.2018YFB1004401) and NSFC under the grant No. 61532021, 61772537, 61772536, 61702522, the Research Funds of Renmin University of China (15XNLQ06) and the Research Funds of Online Education (2017ZD205).

*Cuiping Li is the corresponding author.

References

Anderson, A.; Huttenlocher, D.; Kleinberg, J.; and Leskovec, J. 2014. Engaging with massive online courses. In *WWW'14*, 687–698.

Barto, A. G., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems* 13(1-2):41–77.

Davidson, J.; Liebald, B.; Liu, J.; Nandy, P.; Van Vleet, T.; Gargi, U.; Gupta, S.; He, Y.; Lambert, M.; Livingston, B.; et al. 2010. The youtube video recommendation system. In *Recommender Systems'10*, 293–296.

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data. In *AAAI'18*, 5779–6786.

Ghavamzadeh, M., and Mahadevan, S. 2003. Hierarchical policy gradient algorithms. *Computer Science Department Faculty Publication Series* 173.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI'17*, 1725–1731.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW'17*, 173–182.

He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.-G.; and Chua, T.-S. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based recommendations with recurrent neural networks. In *ICLR'16*.

Huang, J.; Zhao, W. X.; Dou, H.; Wen, J.-R.; and Chang, E. Y. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR'18*, 505–514.

Jing, X., and Tang, J. 2017. Guess you like: course recommendation in moocs. In *WI'17*, 783–789.

Kabbur, S.; Ning, X.; and Karypis, G. 2013. Fism: factored item similarity models for top-n recommender systems. In *SIGKDD'13*, 659–667.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* (8):30–37.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017. Neural attentive session-based recommendation. In *CIKM'17*, 1419–1428.

Narasimhan, K.; Yala, A.; and Barzilay, R. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP'16*.

Parr, R., and Russell, S. J. 1998. Reinforcement learning with hierarchies of machines. In *NIPS'98*, 1043–1049.

Qi, Y.; Wu, Q.; Wang, H.; and Sun, M. 2018. Bandit learning with implicit feedback. In *NIPS'18*.

Qiu, J.; Tang, J.; Liu, T. X.; Gong, J.; Zhang, C.; Zhang, Q.; and Xue, Y. 2016. Modeling and predicting learning behavior in moocs. In *WSDM'16*, 93–102.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI'09*, 452–461.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW'10*, 811–820.

Rendle, S. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology* 3(3):57.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW'01*, 285–295.

Smith, B., and Linden, G. 2017. Two decades of recommender systems at amazon. com. *IEEE Internet Computing* 21(3):12–18.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS'00*, 1057–1063.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181–211.

Tan, Y. K.; Xu, X.; and Liu, Y. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 17–22.

Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; and Su, Z. 2008. Arnetminer: extraction and mining of academic social networks. In *SIGKDD'08*, 990–998.

Wang, L.; Zhang, W.; He, X.; and Zha, H. 2018a. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *SIGKDD '18*, 2447–2456.

Wang, Z.; Liu, J.; Xiao, X.; Lyu, Y.; and Wu, T. 2018b. Joint training of candidate extraction and answer selection for reading comprehension. In *ACL'18*, 1715–1724.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.

Zhang, W.; Du, T.; and Wang, J. 2016. Deep learning over multi-field categorical data. In *ECIR'16*, 45–57.

Zhang, T.; Huang, M.; and Zhao, L. 2018. Learning structured representation for text classification via reinforcement learning. In *AAAI'18*, 6053–6060.