Concept-Aware Deep Knowledge Tracing and Exercise Recommendation in an Online Learning System

Fangzhe Ai
School of Electronics and Information Engineering
Beijing Jiaotong University
17125001@bjtu.edu.cn

Yishuai Chen
School of Electronics and Information Engineering
Beijing Jiaotong University
yschen@bjtu.edu.cn

Yuchun Guo
School of Electronics and Information Engineering
Beijing Jiaotong University
ycguo@bjtu.edu.cn

Yongxiang Zhao
School of Electronics and Information Engineering
Beijing Jiaotong University
yxzhao@bjtu.edu.cn

Guowei Fu
TAL Education Group Inc.
Beijing, China, 100080
fuguowei@100tal.com

Zhenzhu Wang
School of Electronics and Information Engineering
Beijing Jiaotong University
18120144@bjtu.edu.cn

ABSTRACT

Personalized education systems recommend learning contents to students based on their capacity to accelerate their learning. This paper proposes a personalized exercise recommendation system for online self-directed learning. We first improve the performance of knowledge tracing models. Existing deep knowledge tracing models, such as Dynamic Key-Value Memory Network (DKVMN), ignore exercises' concept tags, which are usually available in tutoring systems. We modify DKVMN to design its memory structure based on the course's concept list, and explicitly consider the exercise-concept mapping relationship during students' knowledge tracing. We evaluated the model on the 5th grade students' math exercising dataset in TAL, one of the biggest education groups in China, and found that our model has higher performance than existing models. We also enhance the DKVMN model to support more input features and obtain higher performance. Second, we use the model to build a student simulator, and use it to train an exercise recommendation policy with deep reinforcement learning. Experimental results show that our policy achieves better performance then existing heuristic policy in terms of maximizing the students' knowledge level. To the best of our knowledge, this is the first time that deep reinforcement learning has been applied to personalized mathematic exercise recommendation.

1. INTRODUCTION

Online self-directed learning systems, such as Massive Open Online Courses (MOOCs), are prevailing. These systems, however, assign same exercises to all students, which is inefficient. For comparison, personalized exercises recommendation can improve the efficiency of students' learning. In this paper, we propose a personalized exercise recommendation system for an online self-directed learning service. The system consists of two parts:

- A student knowledge tracing model, which traces a student's knowledge state and predicts whether or not she can finish the exercise correctly.
- A personalized exercise recommendation policy which recommends appropriate exercises to students to accelerate her learning process.

Existing deep knowledge tracing models [9, 13] ignore exercises' knowledge concept properties, which are usually available in tutoring systems. For comparison, in this paper, we propose a concept-aware deep knowledge tracing model. The model is inspired by Dynamic Key-Value Memory Network (DKVMN) model [13]. DKVMN model has a static matrix called *key* which stores the latent knowledge concepts and a dynamic matrix called *value* which stores a student's concept mastery levels. The model computes the correlation between an exercise and the latent concepts in the *key*, and then uses it to read the student's concept mastery levels in the *value*, and predict whether the student will finish the exercise correctly. We improve the DKVMN model as follows: 1) we design its memory structure based on the course's concept list and explicitly consider the exercise-concept mapping relationship during students' knowledge tracing. 2) We enhance it to support more input features, including exercise difficulty, stages, and student practice time. We evaluated the model on the 5th grade students' math exercising dataset in TAL, and found that our model has higher performance than existing deep knowledge tracing models.

In terms of personalized exercise recommendation policy, most of existing algorithms are heuristic, e.g., exercises which are too easy or too hard for a student should be avoided. These algorithms may be not optimal, as they only consider the short-term reward. In this paper, we build a student simulator with our concept-aware deep knowledge tracing model, and then use it to train a flexible and scalable personalized exercise recommendation policy with deep re-

inforcement learning, which considers long-term reward of recommended items.

In summary, the main contributions of this paper are two folds:

- We propose a new exercise-level deep knowledge tracing model whose structure is built based on the course's concept list, and the exercise-concept mapping relationships are utilized during students' knowledge tracing. The model supports more input features and obtains higher performance compared with existing models.
- We propose an exercises recommendation algorithm which uses model-free reinforcement learning with neural network function approximation to learn an exercise recommendation policy. The policy directly operates on raw observations of a student's exercise history. Experimental results show that our policy achieves better performance than existing heuristic policy in terms of maximizing students' knowledge level.

2. RELATED WORK

Knowledge Tracing: The work in [3] proposed a Bayesian-based knowledge tracing model. It models a student's status of a knowledge concept as a binary variable, and updates the probability of her mastering the concept according to her results of doing exercises through a Hidden Markov Model. This model is at the concept level, and ignores the relationship between different concepts. The work in [9] proposed a deep knowledge tracing (DKT) model with recurrent neural network. It models a student's knowledge states as latent variables, and gets better performance than Bayesian-based model does [6]. The work in [12] proposed to improve DKT by considering exercises' semantic features. The work in [13] tried to model the correlation between different latent concepts. Inspired by DKVMN, this paper proposes a model whose structure is explicitly built based on the course's concept list, and the exercise-concept mapping relationship is utilized in the model.

Exercise Recommendation: The work in [1] proposed that a student is recommended by an exercise, if the probability of her doing the exercise correctly is around 50%. The problem of this algorithm is that the threshold 50% is heuristic and may be not optimal. The work in [2] allows experts to specify a ZPD (Zone of Proximal Development) based on current knowledge state of a student, and then chooses the most profitable exercise by multi-armed bandits algorithm. The algorithm can discover the characteristics of students through exploration but it is inefficient, because every student needs an independent exploration process. The work in [5] leverages a DKT model towards recommendation, and frame the problem space using ZPD explicitly facilitated by the DKT model. The work in [7] first estimates each student's knowledge profile from their previous exercise results using SPARFA framework. Then, it uses these knowledge profiles as contexts and applies contextual bandits algorithm to recommend exercises, for maximizing a student's immediate success, i.e., her performance on the next exercise. The problem of this algorithm is that it only considers the next step and thus its performance may be not optimal. The work in [10] evaluated a review scheduling algorithm for spaced repetition systems based on deep reinforcement learning. We are inspired by this work and evaluate the performance of deep reinforcement learning in our math self-directed learning system.

3. BACKGROUND

In this section, we introduce our online learning system and dataset.

3.1 Intelligent Practice System (IPS)

IPS is an online self-directed learning system developed by TAL Education Group, Inc. of China. In IPS, each course (e.g., the 5th grade math) has tens of units. Each unit includes 7 stages, i.e., 1) warming-up exercises before class, 2) in-class exercises before lecture, 3) video lecture, 4) in-class exercises after lecture, 5) homework exercises, 6) unit review exercises, 7) multi-units review exercises. In these 7 stages, stages 1, 2, 3, 4, 5 include contents of a single knowledge concept, but stages 6 and 7 include exercises of other knowledge concepts in order to review. As IPS is a self-directed learning system, a student can choose any teaching unit to study. In a unit, she can also exit current stage or the whole unit at any time. The system records the student's learning duration in each stage, the exercises she practices, and her results, i.e., whether or not the answer is correct.

In IPS, each exercise has three knowledge concept tags, which are provided by experts. The knowledge concept tags have a hierarchical tree structure. For instance, for one exercise, its 1st, 2nd, and 3rd level concept tags are "Number Theory", "Prime Number and Composite Number", and "Decomposition of Prime Factor", respectively.

3.2 Data Set and Data Pre-Processing

We use a sample of anonymized student usage interactions from the 5th grade math curriculum in IPS. We choose exercising records whose first-level knowledge concept is "Number Theory", which has 7 second-level knowledge concepts and 15 third-level knowledge concepts. We further choose students whose exercise records include at least 5 exercises. The resulting dataset includes 44,128 exercise records of 7,124 students.

4. KNOWLEDGE TRACING MODEL

We now introduce our knowledge tracing model based on DKVMN model, and highlight our improvement in aspects of memory structure, knowledge concept weight, and read and update process.

4.1 Concept-Aware Memory Structure

We modify DKVMN to design its memory structure based on the course's concept list. Fig. 1 plots the model's structure, which is based on DKVMN model [13]. As shown in Fig. 1, $\mathbf{M}_t^k$ is the concept embedding matrix whose size is $M \times N$, where N is the number of memory locations, and M is the vector size at each location. We set N equal to the number of the course's knowledge concepts. As we have 1 first-level knowledge concept, 7 second-level knowledge concepts and 15 third-level knowledge concepts, we have $N = 23$. Then, in each location, the student's state for the corresponding knowledge concept is saved. Thus, the

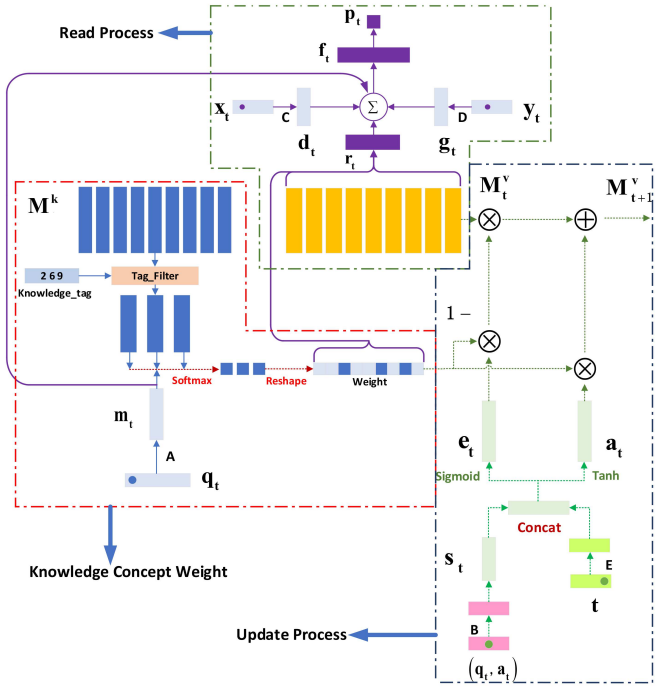

Figure 1: Concept-aware DKVMN model structure.

model's memory architecture is explicitly designed to represent knowledge concepts. For comparison, N is a model parameter in DKVMN representing the number of latent knowledge concepts, e.g., $N = 5$. As our model is inspired by DKVMN, we name it Concept-Aware DKVMN, i.e., *DKVMN-CA*.

4.2 Knowledge Concept Weight

As a student's state of a knowledge concept is saved in the corresponding memory location, when a new exercise arrives, only the exercise's related concepts' memory locations are retrieved and updated. We now present the details of such a procedure. In this section, we calculate the knowledge concept weight (KCW) of the exercise. The weights will be used to calculate the weighted sum of a user's current knowledge concept states to predict her performance on the exercise. It will also be used to update the student's knowledge state after obtaining the answer result of the student on the exercise.

We first obtain the embedding of the arrived exercise. As shown in Fig. 1, when an exercise $\mathbf{q}_t$ arrives at time t, it is first transformed into an embedding vector $\mathbf{m}_t$ through an exercise embedding matrix $\mathbf{A}$. We then calculates the KCW through Algorithm 1. As shown in Algorithm 1, at line 2, we initialize the weight list R. As each exercise has three knowledge concepts, the length of R is 3. Then, for each knowledge concept k (line 2), we calculate the dot product of the embedding of the exercise (i.e., q_t) and the concept embedding (line 3). We then calculate the KCW by obtaining softmax of R, with $Softmax(z_i) = e^{z_i} / \sum_{j=1}^{N} e^{z_j}$ (line 6). Then, we initialize an all-zero vector $Weight$ whose length is the number of the knowledge concepts N(line 7). For each knowledge concept k of the exercise (line 8), we set its weight value in $Weight$.

In summary, DKVMN computes the relationship weights between the exercise and all latent knowledge concepts, but we just compute the relationship weights between the exercise and its knowledge concepts. For the exercise's relationship weights with other concepts, we set them zeros.

Algorithm 1 Knowledge Concept Weight Calculation

Input:
$\mathbf{q}_t$: embedding of the exercise arrived at time t
K_t: knowledge concept list of q_t
$\mathbf{M}^k$: the concept embedding matrix
Output:
$Weight$: Knowledge concept weight of the exercise arrived at time t

/* *Calculate KCW* */
1: $R \Leftarrow [\,]$
2: **for** each $n \in K_t$ **do**
3: $\quad corr \Leftarrow \mathbf{m}_t^T \cdot \mathbf{M}^k[n]$
4: $\quad R.append(corr)$
5: **end for**
6: $R_s \Leftarrow Softmax(R)$
/* *Reshape the weight vector to make its length equal to the number of concepts*/
7: $Weight \Leftarrow [0, ..., 0]$
8: $i \Leftarrow 0$
9: **for** $i < 3$ **do**
10: $\quad \text{Weight}[K_t[\text{i}]] \Leftarrow R_s[i]$
11: $\quad i \Leftarrow i + 1$
12: **end for**
13: **return** $Weight$

4.3 Read Process

We then use the obtained KCW to calculate the weighted sum of the user's current knowledge concept states to predict the student's performance on the exercise. Denote KCW by $\mathbf{w}$, we have $\mathbf{r}_t = \sum_{i=1}^{N} \mathbf{w}_i \mathbf{M}_t^v$, i.e., the knowledge state of concepts related to the exercise q_t.

We further concatenate $\mathbf{r}_t$ with the embeddings of the exercise's difficulty and stage feature, i.e., $\mathbf{d}_t$ and $\mathbf{g}_t$. The result then passes through a fully connected layer with activation function Tanh to get a summary vector $\mathbf{f}_t$, which contains all information of the student's knowledge state related to $\mathbf{q}_t$ and the exercise's features, i.e.,:

$$\mathbf{f}_t = Tanh(\mathbf{W}_0^T[\mathbf{r}_t, \mathbf{d}_t, \mathbf{g}_t, \mathbf{m}_t])$$

where $Tanh(z_i) = (e^{z_i} - e^{-z_i})/(e^{z_i} + e^{-z_i})$.

Finally, $\mathbf{f}_t$ passes through a fully connected layer to output the probability that the student would do the exercises $\mathbf{q}_t$ correctly. Denote the probability by $\mathbf{p}$, we have

$$\mathbf{p} = Sigmoid(\mathbf{W}_1^T \mathbf{f}_t)$$

where $Sigmoid(z_i) = 1/(1 + e^{-z_i})$.

4.4 Update Process

We then use the KCW to update the student's knowledge state after observing the her answer result. The update process updates the value matrix $\mathbf{M}_t^v$, which represents the student's current state of knowledge concept k. Our model is different from DKVMN model in that we consider the student's exercising duration in the update process. For

comparison, DKVMN ignores this student behavior feature. Specifically, the work in [8] proposed that the a student's duration of solving a problem is related to her master level of latent problem solving skills. Inspired by this work, in our model, after a student finishes an exercise, her answer result (i.e., correct or wrong) $\mathbf{a}_t$ and exercising duration are used to update $\mathbf{M}_t^v$. Because the exercising duration is a continuous variable, it is firstly discretized according to its distribution and then represented by its embedding $\mathbf{t}$. We then concatenate $\mathbf{t}$ with the joint embedding $\mathbf{s}_t$ of the answer vector $(\mathbf{q}_t, \mathbf{a}_t)$, to update $\mathbf{M}_t^v$, as shown in Fig. 1.

The other update process is same as that of DKVMN. It includes erase subprocess and add subprocess. Erase vector is computed as $\mathbf{e} = Sigmoid(\mathbf{E}^T[\mathbf{s}_t, \mathbf{t}])$, where $\mathbf{E}$ is the erase weights. Add vector is computed as $\mathbf{a} = Tanh(\mathbf{D}^T[\mathbf{s}_t, \mathbf{t}])$, where $\mathbf{D}$ is the add weights. Then the new memory matrix $\mathbf{M}_{t+1}^v$ is computed by

$$\mathbf{M}_{t+1}^v(i) = \mathbf{M}_t^v(i)[1 - \mathbf{w}(i)\mathbf{e}][1 + \mathbf{w}(i)\mathbf{a}]$$

The parameters of the model are learned by minimizing a standard cross entropy loss between the predicted user answer result p_t and her true result y_t:

$$L = -\sum_t ((y_t log p_t) + (1 - y_t) log(1 - p_t))$$

In summary, compared with DKVMN, we design the model structure based on the course's concept list, and then explicitly consider the exercise-concept mapping relationship and other exercise's features during students' knowledge tracing.

5. REINFORCEMENT LEARNING BASED EXERCISES RECOMMENDATION

Based on the DKVMN-CA student knowledge tracing model, we build a student simulator which provides environment for reinforcement learning, and train a personalized exercise recommendation agent with deep reinforcement learning method.

Similar to [10], we model the recommendation process as a Partially Observable Markov Decision Process (POMDP), where the model state is the student's latent knowledge state and the action is the recommendation of an exercise. At time t, the reinforcement learning agent cannot observe the student's latent knowledge state s_t. Instead, it can observe the student's exercise and answer result (i.e., correct or wrong) o_t which is conditioned on the latent knowledge state $p(o_t|s_t)$. Thus, at time t, the agent needs to recommend an exercise a_t based on the student's exercising history before t, which is denoted by h_t. We have $h_t = (o_1, a_1, o_2, a_2, \ldots, o_{t-1}, a_{t-1})$. After the student finishes the recommended exercise a_t, her latent knowledge state will turn to s_{t+1} by a transition function $p(s_{t+1}|s_t, a_t)$.

We define the reward r_t of an action a_t as

$$r_t = \frac{1}{K}\sum_{i=1}^{K} P_{t+1}(q_i), \quad (1)$$

where K is the number of exercises , and $P_{t+1}(q)$ denotes the probability of the student getting exercise q correct after finishing the recommended exercise at state s_{t+1}. It is predicted by the student simulator. So, we name it as the student's *Predicted Knowledge*.

The purpose of optimization is to maximize the reward R of policy π:

$$R = \mathbb{E}_\tau[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t)],$$

where trajectories $\tau = (s_1, o_1, a_1, s_2, o_2, a_2, ...)$ are drawn from the trajectory distribution induced by policy $\pi : p(s_1)$ $p(o_1|s_1)\pi(a_1|h_1)p(s_2|s_1, a_1)p(o_2|s_2)\pi(a_2|h_2)$.... Thus, as for the action-value function Q^π, the reward of the recommended exercise sequence at t is:

$$Q^\pi(h_t, a_t) = \mathbb{E}_{s_t|h_t}[r_t(s_t, a_t)] + \mathbb{E}_{\tau>t|h_t,a_t}[\sum_{i=1}^{\infty} \gamma^i r(s_{t+i}, a_{t+i})]$$

where $\tau > t = (s_{t+1}, o_{t+1}, a_{t+1}...)$ is the future trajectory. The algorithm then recommends the exercise $q_{'}$ which has the maximal reward, i.e., $q_{'} = \mathbf{max}_a Q^\pi(h_t, a)$. Similar to [10], we approximately solve the POMDP using Trust Region Policy Optimization (TRPO) algorithm [11], with an off-the-shelf implementation from rllab [4].

6. PERFORMANCE EVALUATION

In this section, we present the performance evaluation results of our system.

6.1 DKVMN-CA Knowledge Tracing Model

We evaluated our model on our IPS dataset. To evaluate it, we conducted 50 experiments. In each experiment, we randomly split the users into two groups: training users and testing users. Their percentages are 70% and 30%. We then trained the model with the training users and evaluated the model on the testing users. Similar to [9], we use area under the curve (AUC) as the performance metric. We report the maximal, mean, and the standard deviation of the testing users' AUCs of all 50 experiments.

6.1.1 *Efficiency of Concept-Aware Design*

We first report the efficiency of designing the model's architecture based on the course's knowledge concepts. We compare the performance of DKVMN and DKVMN-CA without the help of other input features, including exercise difficulty, stage, and duration. The results are shown in Table. 1. See the rows "DKVMN" and "DKVMN-CA". As shown in Table. 1, our model obtains an AUC of 0.724, which is higher than that of DKVMN model, i.e., AUC = 0.712. Such an improvement is considerable, considering the small improvement DKVMN provides over the DKT baseline (AUC = 0.711). Such a result means that the design of the model's architecture based on the course's knowledge concepts is efficient.

To highlight the necessity of our design, we also evaluate another method which also uses exercises' knowledge concept tags, i.e., represents a knowledge concept by its embedding and then concatenate it with the embedding of the exercise. Its' performance is shown in Table. 1. See the row "DKVMN-KC". As shown in Table 1, its mean AUC is 0.714, meaning a very small improvement over the DKVMN (AUC = 0.712). Thus, it is necessary to design the model's

Table 1: AUC of Models with Different Features

Model	Mean AUC	Max AUC	Variance
LSTM	0.711	0.712	1.86e-05
DKVMN	0.712	0.720	2.05e-05
DKVMN-KC	0.714	0.724	1.85e-05
DKVMN-CA	0.724	0.731	2.14e-05
DKVMN-CA + Stage	**0.728**	0.736	**1.48e-05**
DKVMN-CA + Duration	0.725	0.737	1.75e-05
DKVMN-CA + Difficulty	0.726	0.736	2.44e-05
DKVMN-CA + Stage, Duration	0.726	**0.739**	2.43e-05

architecture based on the course's knowledge concepts to fully utilize their capacity to improve the model.

6.1.2 *Efficiency of Other Exercise Features*

We then evaluate the efficiency of adding other features, including exercise difficulty, stage, and duration. The results are shown in Table 1. See the rows "DKVMN-CA + Difficulty", "DKVMN-CA + Stage", and "DKVMN-CA + Duration". As shown in Table 1, these features can further improve the model's performance. For example, the mean AUC of "DKVMN-CA + Stage" is 0.728, which is higher than that of DKVMN model (AUC = 0.712).

6.2 Exercises Recommendation

6.2.1 *Evaluation of Students' Knowledge Growth Process*

We use the Expectimax algorithm proposed in [9] as the baseline algorithm. In the Expectimax algorithm, the system first calculates a student's predicted knowledge assuming an exercise is recommended to the user to practice. It then chooses the exercise with the highest predicted knowledge to recommend.

To compare the two algorithms, we first randomly pick 15 students in our dataset. For each student, we conduct two experiments, one experiment for one algorithm. In each experiment, we first initialize the student simulator using the student's historical practice sequence. Then, we continuously recommend 50 exercises to the student simulator with the recommendation algorithm. During the process, we record the average of the 15 students' predicted knowledge as Eq.(1) at each step of recommendation. The results are shown in Fig. 2. As shown in Fig. 2, the students served by RL policy has a higher mean predicted knowledge than the students served by the Expectimax policy after 50 exercises. Moreover, after about 10 exercises, the mean predicted knowledge of the students served by the Expectimax policy stops increasing, meaning that the policy cannot find exercises which can help the students to improve the performance any more. For comparison, served by the RL policy which considers long-term reward of action, the students' mean predicted knowledge keeps increasing, meaning that the RL policy still can find exercises which can help the student to improve performance.

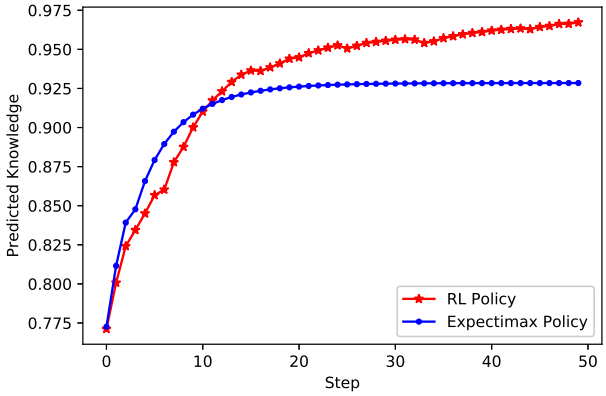

Figure 2: Performance evaluation results

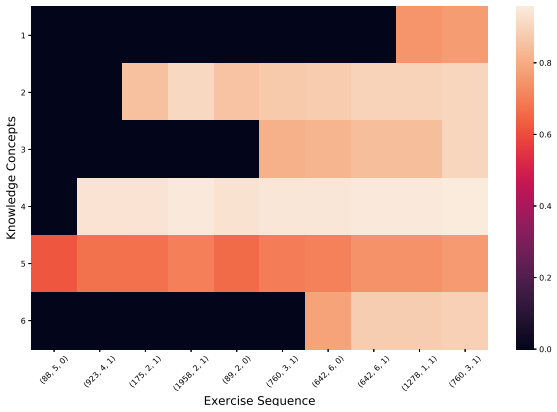

Figure 3: Knowledge State Variation

6.2.2 *Evaluation of Recommendation Process*

We design another experiment to observe the recommendation behavior of the RL policy. We randomly pick a student who has practiced five exercises, and use her exercise sequence to initialize the student simulator. Then, we serve her with five more exercises using the RL recommendation policy. Fig. 3 shows the results. The x label of Fig. 3 shows the 10 exercises' IDs, concepts, and results. For example, the first record (88, 5, 0) means that the exercise ID is 88, which is related to concept 5, and the student's answer is wrong. As the 10 exercises are related to 6 concepts, we plot the student's predicted knowledge of each concept in Fig. 3. For instance, as the student fails in the first exercise 88, which is related to the 5th concept, the student's knowledge status on the 5th concept is relatively low. The status of the knowledge concepts not covered by the student's history exercises are indicated in black. We now observe the recommended exercises. We have the following observations:

- As shown in Fig. 3, the first five exercises are related to concepts 2,4,5, and the later five exercises are related to concepts 1,3,6, suggesting the RL algorithm wants to explore the student's capacity in other concepts.

- After the student succeeds in exercise 760, which is related to the concept 3 (Decomposition of Prime Factor), the algorithm recommends the exercise 642, which is related to concept 6 (Maximum common factor and Least common multiple). As concept 6 is related to concept 3, such a recommendation is reasonable.

- The student, however, fails to finish the exercise 642. Thus, the algorithm recommends exercise 642 again. This time, the student succeeds to finish it, meaning that the model captures the phenomenon during training that a student who failed in exercise 642 may succeed if she retries. Such a result is interesting.

- After the student succeeds in exercise 642, which is related to concept 6 (Maximum common factor and Least common multiple), the model's estimation of the student's capacity on concept 3 (Decomposition of Prime Factor)also slightly increases. As these two concepts are indeed related, such a result is reasonable.

- Then, the algorithm turns to another concept again, i.e., it recommends the exercise 1278, which is related to concept 1. While the student succeeds in the exercises, the estimated student's knowledge status on the concept 1, however, is relatively low, suggesting that the exercise is relative easy.

- At last, the exercise 760 is recommended again, and the student succeeds in it. As a result, the model's estimation of the student's capacity on concept 3 increases, suggesting that reviewing is beneficial for study.

7. CONCLUSION

In this paper, we improve DKVMN by designing its neural network structure based on a course's concept list, and explicitly considering the exercise-concept mapping relationship during students' knowledge tracing. We also enhance the DKVMN model to consider more input features. Experimental results show that our model has higher performance than existing deep knowledge tracing models. We also propose an exercises recommendation algorithm which uses model-free reinforcement learning with neural network function approximation to learn an exercise recommendation policy that directly operates on raw observations of a student's exercise history. Our experimental results demonstrate that our policy achieves better performance than existing heuristic policy in terms of maximizing the students' knowledge level. To the best of our knowledge, this is the first time that deep reinforcement learning has been applied to personalized mathematic exercise recommendation.

8. ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China under Grants 61572071, 61872031, and 61301082.

9. ADDITIONAL AUTHORS

Additional authors: Guangyan Wang (College of Education, Hebei University, email: gywang@163.com).

10. REFERENCES

[1] I.-A. Chounta, B. M. McLaren, P. L. Albacete, P. W. Jordan, and S. Katz. Modeling the zone of proximal development with a computational approach. *EDM*, 2017:56–57, 2017.

[2] B. Clement, D. Roy, P.-Y. Oudeyer, and M. Lopes. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*, 2013.

[3] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[4] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[5] W. Jiang, Z. A. Pardos, and Q. Wei. Goal-based course recommendation. pages 36–45, 2018.

[6] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

[7] A. S. Lan and R. G. Baraniuk. A contextual bandits framework for personalized learning action selection. In *EDM*, pages 424–429, 2016.

[8] R. Pelánek and P. Jarušek. Student modeling based on problem solving times. *International Journal of Artificial Intelligence in Education*, 25(4):493–519, 2015.

[9] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. volume 3, pages págs. 19–23, 2015.

[10] S. Reddy, S. Levine, and A. D. Dragan. Accelerating human learning with deep reinforcement learning. 2017.

[11] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[12] Y. Su, Q. Liu, Q. Liu, Z. Huang, Y. Yin, E. Chen, C. Ding, S. Wei, and G. Hu. Exercise-enhanced sequential modeling for student performance prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774. International World Wide Web Conferences Steering Committee, 2017.