World Wide Web (2022) 25:1343–1372
https://doi.org/10.1007/s11280-022-01041-2

Multi-type factors representation learning for deep learning-based knowledge tracing

Liangliang He[1] · Jintao Tang[1] · Xiao Li[2] · Pancheng Wang[1] · Feng Chen[1] · Ting Wang[1]

Received: 13 September 2021 / Revised: 14 February 2022 / Accepted: 8 March 2022 /
Published online: 3 May 2022

Abstract
Knowledge Tracing (KT) refers to the problem of predicting future learner performance given their historical interactions with e-learning platforms. Recent years, Deep Learning-based Knowledge Tracing (DLKT) methods show superior performance than traditional methods due to their strong representational ability. However, researchers usually focus on innovations in model structure, while ignoring the importance of Representation Learning (RL) for DLKT. Investigating previous studies, it is found that the mining and integration of learning-related factors can effectively improve the performance of DLKT models. This paper focuses on providing a model embedding interface for DLKT by considering multiple types of learning-related factors. We first explore and analyze four types of learning-related factors: exercise and skill, the attributes of exercise, learners' historical performance, and learners' forgetting behavior in the learning process. We then propose an Extensible Representation Learning (ERL) approach for DLKT to extract and integrate the representations of these four types of factors by setting five components: base embedding, auxiliary embedding, performance embedding, forgetting embedding, and embedding integration. Finally, we apply ERL into two mainstream DLKT models and comprehensively evaluate the proposed approach on several real-world benchmark datasets. Results show that ERL can significantly improve the performances of these two network on predicting future learner responses.

Keywords Knowledge tracing · Deep learning · Representation learning · Learner modeling

1 Introduction

With the popularity of e-learning platforms, learners can acquire knowledge by self-study without leaving home. To recommend suitable learning contents to learners, e-learning platforms need to understand learners' knowledge accurately [24], which is often done with Knowledge Tracing (KT). KT is an important task in e-learning. For example, it is

✉ Ting Wang
tingwang@nudt.edu.cn

Extended author information available on the last page of the article

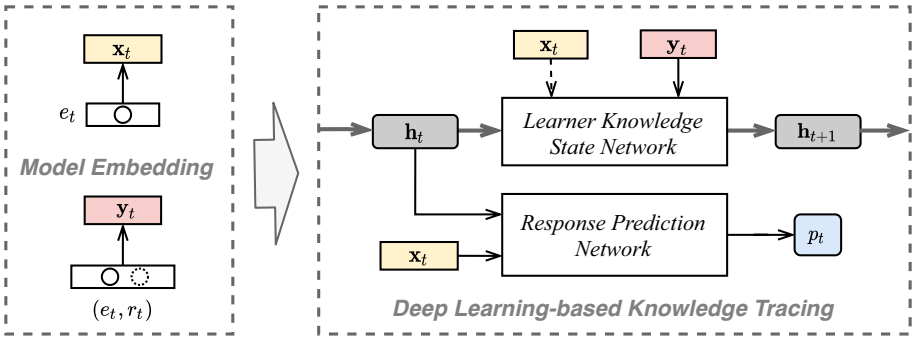


Fig. 1 General working paradigm of DLKT (only the workflow at timestamp t is shown, and the following model embedding and knowledge tracing are default at timestamp t)

a stepping stone for the tasks of the knowledge graph [22, 32]. The goal of KT task is to model the Knowledge State (KS) of each learner, i.e., the level of the learner's mastery of skills, based on the history of the learner's interactions with the platform. On an e-learning platform, learners can learn related skills by completing specific exercises (e.g., if *addition* is a skill, "1 + 1" is its exercise.), and the system traces the learner's KS about the learning skills based on a KT model. Finally, the platform determines whether the learner have mastered these skills by a when-to-stop policy [31].

Usually, KT is formulated as a supervised sequence learning problem [38]: given a learner's interaction sequence $\mathbf{I}_{t-1} = (\mathbf{i}_1,\mathbf{i}_2 \ldots \mathbf{i}_{t-1})$ up to the timestamp t (where $\mathbf{i} = (e,r)$ is an input pair containing the exercise e at one timestamp and the learner's response r (correct/incorrect) to e), the exercise e_t at timestamp t on a specific learning scenario, KT models try to predict the probability that the learner will correctly perform a learning action (e.g., responding the exercise) at timestamp t, i.e., $p(r_t = 1|e_t,\mathbf{I}_{t-1})$, [13, 16, 36].

Recent years, Deep Learning-based Knowledge Tracing (DLKT) methods [11, 30, 36, 38] have shown superior performance than traditional models, such as Bayesian knowledge tracing [7], latent factor models [4, 28] and item response theory [15, 33]. Figure 1 shows the general working paradigm of DLKT, where the role of *Model Embedding* is to provide the exercise embedding (as $\mathbf{x}$), the other is the exercise-response embedding (as $\mathbf{y}$) for DLKT. The former takes part in the prediction process in *Response Prediction network* together with the current KS of the learner, while the latter is used to update the learner's current KS in *Leaner Knowledge State Network*, and the updated KS is used to predict the response of the exercise in the next timestamp.

Theoretically, $\mathbf{x}$ and $\mathbf{y}$ are generated based on the exercise tag and the corresponding real response tag, i.e., $\mathbf{x}$ and $\mathbf{y}$ represent the embedding of exercise e and the embedding of exercise-response (e,r), respectively. However, due to the sparsity of exercise data [11, 26], earlier DLKT models [30, 38] use skill embedding instead of exercise embedding as the model input to avoid over-parameterization and over-fitting. As the sparsity of exercise data is relieved to some extent [11, 26], more and more factors (e.g., exercise [11], forgetting [24, 27], exercise text [20], etc) are integrated into the specific model, making exercise a full representation.

However, due to the difference of learning content and learning setting in different e-learning platforms, the types and quantities of learning-related factors modeled in specific models are different, which is not conducive to the subsequent application and

promotion of the model. Therefore, it is necessary to provide a systematic method to guide the representation learning (RL) of these factor, which has not received much attention in DLKT so far. RL [2] makes it easier to extract useful information when building classifiers or predictors by learning representations of the data, which has been successfully applied in various fields of machine learning, such as object recognition [18, 21], natural language processing [3, 14], transfer learning [1, 8] and so on.

In this paper, we propose an extensible representation learning approach, dubbed ERL, which aims to provide a model embedding interface for DLKT by mining and integrating multiple types of learning-related factors. We first emphasize the importance of factor mining and integration for DLKT by investigating the results of recent models integrated multi-factors. Then, we explore and analyze four types of learning-related factors: exercise and skill, the attributes of exercise, learners' historical performance, and learners' forgetting behavior in the learning process, which is inspired by the nature of learning behavior and previous researches. Moreover, we extract the representations of these four types of factors by setting four embedding components: Base Embedding (BE), Auxiliary Embedding (AE), Performance Embedding (PE) and Forgetting Embedding (FE), respectively. BE involves the representation extraction of skill data and exercise data, dealing with the sparsity of exercise data; AE involves the representation extraction of various features (e.g., template, hint, etc.) of the exercise, and provides *local* extensibility to integrate one or more auxiliary factors; PE involves the representation extraction of the historical performance of each learner; FE involves the representation extraction of the forgetting features of each learner, including the lag time between two adjacent interactions with the same exercise and two successive interactions. Finally, we integrate the representations of the above four types of factors by setting a Embedding Integration (EI) component, which can effectively solve the problems of over-parameterization and over-fitting caused by integration of too many factors.

To illustrate effectiveness of four types of learning-related factors and the usability of the final representation learning approach, we apply the proposed approach into two mainstream representative DLKT models: DKVMN and AKT (the latest DLKT network). We design extensive experiments on three real-world datasets to comprehensively evaluate the two applied models. Results show that the proposed approach can significantly improve the performances of the latest network on predicting future learner responses, and the final performances outperform the state-of-the-art DLKT models[1]. A preliminary version of this report appeared in the Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME) [12].

The contributions of this work are summarized as follows:

(1) Investigating the results of recent studies in DLKT, we find that although the structural innovations of the model have been fully demonstrated, the improvement of model performance brought by factor mining should not be underestimated.
(2) Exploring four types of factors that may affect learners' knowledge tracing results, and analyzing these factors at the data level from the perspective of influencing the exercise-making accuracy of learners.

[1] The corresponding source code and all preprocessed datasets are available at https://github.com/HLBilove/ERL-master

(3) Proposing an extensible representation learning approach, dubbed ERL, which is used mined and integrated the above four types of factors, to provide a model embedding interface for DLKT.
(4) Applying ERL two mainstream representative DLKT models, and evaluating the two applied instances of ERL on three real-world benchmark datasets. The results show that the proposed framework improves state-of-the-art KT methods on predicting future learner responses.

In the remainder of this paper, we introduce the related work in the next section. Section 3 investigates the role of model innovation and factor mining in DLKT. The ERL approach is proposed in Section 4; two application instances, ERL+DKVMN and ERL+AKT, are proposed in Section 5. Experiments and Analysis follow in Section 6, with conclusion afterwards in the last section.

2 Related work

Since this paper aims to mine and integrate representations of multi-type learning-related factors to improve DLKT, this section will review the related works in this field in terms of the number of factors integrated in existing models.

2.1 Single-factor models

For single-factor KT models, the single factor usually refers to exercise or skill. The exercise library is considerably larger than the skill set, so many exercises are only learned by few learners in most e-learning platforms [11, 26], resulting in sparse exercise data (Figure 2 shows the distribution of the number of skills and the number of exercises on the learned times in ASSISTments2009 and ASSISTments2017, respectively). Due to the sparsity of exercise data, skills instead of exercises are used to generate $\mathbf{x}$ and $\mathbf{y}$ in DLKT models at the beginning to avoid over-parameterization and over-fitting [11], i.e., $s = e$ in Figure 1 (ps, non-sparse exercise data is still the first choice for $\mathbf{x}$ and $\mathbf{y}$).

The first single-factor DLKT model is Deep Knowledge Tracing (DKT) [30], which applies Long Short-Term Memory (LSTM) network [29] to KT tasks. DKT uses hidden states as a kind of summary of the past exercise-making sequence. Dynamic Key-Value Memory Networks (DKVMN) [38] models the user's KS into two memory matrices: *key*-memory and *value*-memory, which are used to trace the user's KS about each underlying skill by automatically learning the correlation between the input exercise and the skill. The Self-Attentive Knowledge Tracing (SAKT) model [26] applies attention mechanism to DLKT for the first time, to deal with the sparsity of the exercise data. SAKT predicts the user's performance on the current exercise by considering the relevant exercises from his/her past exercise-making sequence.

2.2 Double-factor models

For double-factor KT models, the two factors usually refers to exercise and skill. Although skill data in single-factor models circumvent the sparsity of exercise data effectively, $\mathbf{x}$ and $\mathbf{y}$ to ignore the differences of exercises covering one same skill [11]. Thus, both skill data

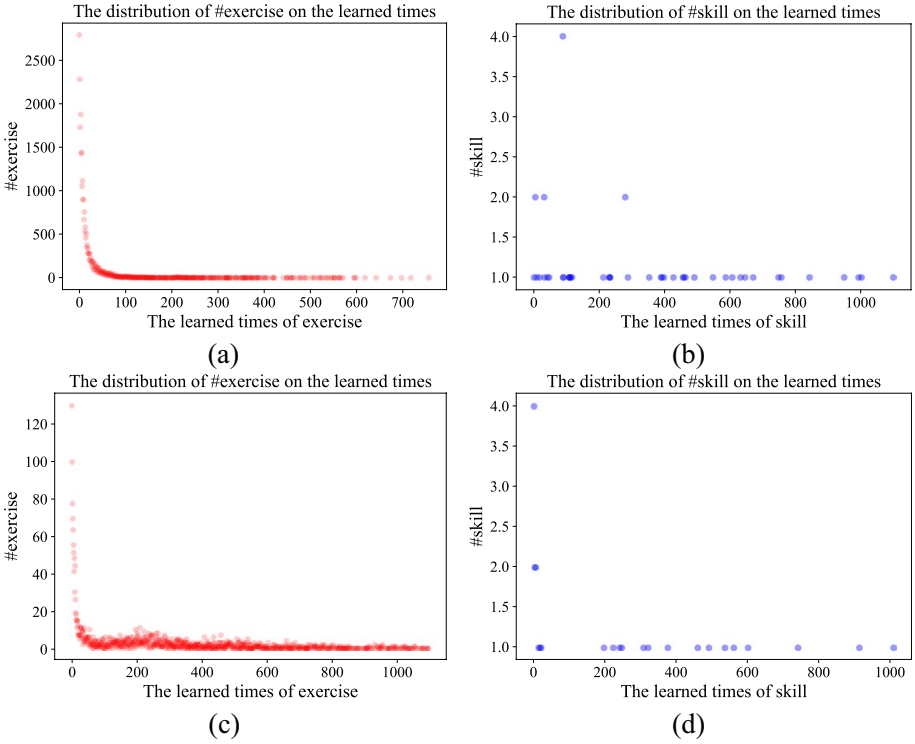


Fig. 2 Distributions of #exercise and #skill on the learned times in ASSISTments2009 (of (**a**) and (**b**)) and ASSISTments2017 (of (**c**) and (**d**)). Where the exercise data is very sparse, because there are very few exercises that have been learned more often, and on the contrary, skills are learned more evenly

and exercise data are used together to generate $\mathbf{x}$ and $\mathbf{y}$ in the double-factor models, resulting in significant performance gains.

Wang et al. [35] propose a novel Deep Hierarchical Knowledge Tracing (DHKT) model exploiting the hierarchical relations between skills and exercises, which are modeled by the hinge loss on the inner product between the average embedding of all skills covering one single exercise and the exercise embedding. Unfortunately, DHKT ignores the sparsity of the exercise data, when embedding the hierarchical relations.

Nagatani et al. [24] extends DKT by modeling the data related to forgetting. They consider both the learner's exercise-making sequence and the different forgetting behaviors in the process of exercise-making, and the latest extension model is called Bi-Interaction Deep Knowledge Tracing (BIDKT) [17].

Ghosh et al. [11] propose a novel Attentive Knowledge Tracing (AKT) model which is completely dependent on attention network. AKT improves upon existing KT methods by proposing a new monotonic attention mechanism to summarize past user performance. In addition, they propose a Rasch Model-based Embedding (RME) method to model embedding, and the embeddings of RME for $\mathbf{x}$ ($\mathbf{x}^{RME}$) and $\mathbf{y}$ ($\mathbf{y}^{RME}$) are as follows:

$$\mathbf{x}^{RME} = \mathbf{f}_s + \mu_e \cdot \mathbf{v}_s, \tag{1}$$

$$\mathbf{y}^{RME} = \mathbf{p}_{(s,r)} + \mu_e \cdot \mathbf{v}_{(s,r)}, \tag{2}$$

where $\mathbf{f}_s \in \mathbb{R}^D$ and $\mathbf{p}_{(s,r)} \in \mathbb{R}^D$ denote the factor embedding and pair embedding of the skill this exercise covers, respectively; $\mathbf{v}_s \in \mathbb{R}^D$ is a factor vector that summarizes the variation in exercises covering this skill; $\mathbf{v}_{(s,r)} \in \mathbb{R}^D$ is a pair vector that summarizes the variation in exercises and their corresponding responses; $\mu_e \in \mathbb{R}$ is a scalar *difficulty* parameter that controls how far this exercise deviates from the skill it covers. However, although this setup in RME alleviates over-parameterization and over-fitting of models to a certain, compared with the multi-dimensional continuous vector, the 1-dimensional continuous scalar can carry very limited information, so the representation of exercises cannot be fully extracted when the exercise data is not sparse.

2.3 Multi-factor models

To improve the performance of DLKT, multiple learning-related factors are integrated into some specific models [20, 24, 27]. Liu et al. propose an Exercise-aware Knowledge Tracing (EKT) framework by integrating the information of skills, exercise-making sequence and the content text of exercises into a single model [20]. Pandey and Srivastava propose a novel Relation-aware self-attention Knowledge Tracing (RKT) model by improving the SAKT model. There are three types of information are combined in RKT, including the exercise-making sequence, the relations between skills and time delay since the last interaction, and the text information of the exercise content [27]. Admittedly, integrating more learning-related factors in a specific DLKT model does improve the performance of the model, but the integration mode of all factors involved depends on the specific DLKT model and is difficult to extend. Therefore, this paper aims to provide a model embedding interface for DLKT by mining and integrating multiple types of learning-related factors.

3 Investigation on model innovation and factor mining

To understand the effect of Model Innovation (MI) and Factor Mining (FM) on the final performance of the models in DLKT, we investigate the experimental results of recent DLKT models integrated multi-factors from two aspects of MI and FM. This section only takes two latest representative works (AKT [11] and RKT [27]) as examples to illustrate the results of our analysis. The datasets involved in this section are from the corresponding specific papers.

3.1 Model innovation

To understand DLKT's achievements in MI in recent years, we extract the partial experimental results in the papers of AKT and RKT, showing in Tables 1 and 2 respectively, where AKT^{Skill} and RKT^{Skill} denote the corresponding variants only integrating the skill factor, respectively, best models are bold and second best models are italic (working on all tables in this paper).

Table 1 shows that, compared with the earliest DLKT model (DKT), the largest performance improvements (from AKT^{Skill}) due to MI are 0.39% (on Statics2011), -0.01% (on ASSISTments2009) and 0.26% (on ASSISTments2017), respectively. Incredibly, the earliest

Table 1 AUC values of AKT and its baselines which only integrate skill factor

Models	ASSISTments2009[a]	ASSISTments2017[b]	Statics2011[c]
DKT	**0.817**	*0.726*	*0.823*
DKVMN	0.809	0.707	0.820
SAKT	0.752	0.657	0.803
AKT^{Skill}	*0.817*	**0.728**	**0.827**

[a]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data.

[b]https://sites.google.com/view/assistmentsdatamining/dataset

[c]https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507

Table 2 AUC values of RKT and its baselines which only integrate skill factor

Models	ASSISTments2012[a]	POJ[b]	Junyi[c]
DKT	0.712	0.656	0.814
DKVMN	0.701	**0.704**	0.822
SAKT	**0.735**	*0.696*	**0.834**
RKT^{Skill}	*0.730*	0.667	*0.830*

[a]https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-withaffect

[b]https://www.junyiacademy.org/

[c]http://poj.org/

Table 3 AUC values of AKT and its baselines which integrate different factors

	ASSISTments2009		ASSISTments2017	
Models	Skill	Skill+Exercise	Skill	Skill+Exercise
DKT	0.817	**0.818**	0.726	**0.754**
DKVMN	0.809	**0.824**	0.707	**0.763**
SAKT	0.752	**0.778**	0.657	**0.714**
AKT	0.817	**0.835**	0.728	**0.770**

model shows second best performance on the whole over all datasets, and performs best on ASSISTments2009.

Table 2 shows that, compared with DKT, the largest performance improvements due to MI are 3.2% (on ASSISTments2012 from SAKT), 7.3% (on POJ from DKVMN) and 2.5% (on Junyi from SAKT), respectively. Incredibly, the earliest model shows second best performance over all datasets. Although the overall improvement is significant, best models are not the latest model (RKT) when only the skill factor is considered, and no single model is optimal over all datasets.

Table 4 AUC values of RKT and its baselines which integrate different factors

Models	ASSISTments2012	POJ	Junyi
RKT	0.730	0.667	0.830
RKT+*Performance*	0.735	0.696	0.834
RKT+*ExerciseText*	0.759	0.759	0.832
RKT+*Forgetting*	0.778	0.788	0.833

3.2 Factor mining

We also extract the partial experimental results in papers of AKT and RKT, which can illustrate the performance improvement caused by FM, showing in Tables 3 and 4, respectively.

Table 3 shows that all models have achieved significant performance improvements on the whole by integrating exercises (E) based on modeling skills (S) over both ASSISTments2009 and ASSISTments2017. The maximum increases (from SAKT) are 3.5% (on ASSISTments2009) and 8.6% (on ASSISTments2017). For the latest network AKT, the increases are 2.2% (on ASSISTments2009) and 5.8% (on ASSISTments2017).

Table 4 shows that, compared with RKT which only models skill data, RKT+*Performance* integrating additional performance data, achieves 0.68%, 4.3% and 0.48% improvement on ASSISTments2012, POJ and Junyi, respectively; RKT+*ExerciseText* integrating additional exercise text data, achieves 4.0%, 1.3% and 0.24% improvement on ASSISTments2012, POJ and Junyi, respectively; RKT+*Forgetting* integrating additional forgetting data, achieves 6.6%, 18.1% and 0.36% improvement on ASSISTments2012, POJ and Junyi, respectively.

3.3 Summary of investigation

Compared with the performance improvement brought by MI, the performance improvement brought by FM is relatively more significant and stable. Therefore, we believe that FM for DLKT should not be neglected and should be given at least the same importance as MI. Unfortunately, there is a lack of systematic methods to guide Representation Learning (RL) for DM in DLKT by far.

Existing DLKT models embed one or more factors to obtain the input of their models. In general, the more extended factors, the better the model performance. However, due to the difference of learning content and learning environment in different online learning, the types and quantities of learning factors integrated into the DLKT models are different and this expansion lacks a clear direction, which is not conducive to the subsequent application and promotion of the model.

This paper focus on providing a model embedding interface for DLKT by learning representations of multi-type learning-related factors. As shown in Figure 1, $\mathbf{x}$ and $\mathbf{y}$ are used to provide factor embedding and pair embedding to the DLKT model. Therefore, our task is to learn $\mathbf{x}$ and $\mathbf{y}$ based on the static data from the e-learning platform (i.e., exercise, skill and etc) and the dynamic data of learner's interaction with the platform (i.e., performance, forgetting and etc).

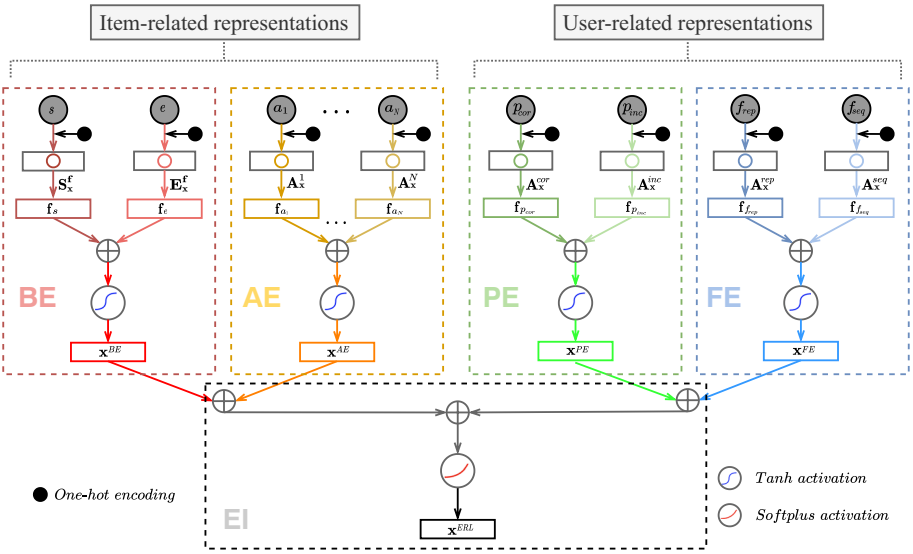


Fig. 3 Architecture of ERL for the factor embedding

4 Extensible representation learning for factor embedding

According to the formal definition in Section 1, the KT task has been abstracted as a prediction problem of unknown response, i.e., predicting a learner's unknown response to a certain exercise at a certain timestamp in the future. Therefore, the model embedding of DLKT should consider the representations of both exercise and learner. For convenience, the former are collectively referred to as Item-related Representations (IR), while the latter are collectively referred to as User-related Representations (UR). In addition to exercise and skill in IR, we believe that other attributes of exercise (e.g., template, hints, etc) should not be ignored; we also believe that learners' historical performance and forgetting behavior in UR will greatly affect their future learning.

In this section, we first analyze the influence of the above four types of factors on learners' learning behavior (exercise-making); then, we extract the representations of the four types of factors by setting four embedding components: Base Embedding (BE), Auxiliary Embedding (AE), Performance Embedding (PE) and Forgetting Embedding (FE), respectively; finally, we integrate the representations of the above four types of factors by setting a Embedding Integration (EI) component to generate the final factor embedding ($\mathbf{x}$) and pair embedding ($\mathbf{y}$) for DLKT. The complete method is called Extensible Representation learning (ERL). Figure 3 shows the architecture of ERL for $\mathbf{x}$ when the exercise data is not sparse. Let D denotes the dimension of all factor and pair embeddings.

4.1 Base embedding

To make both embeddings $\mathbf{x}$ and $\mathbf{y}$ reflect the individual differences among exercises covering the same skill, RME weights the vector embedding of the skill using the scalar *difficulty*

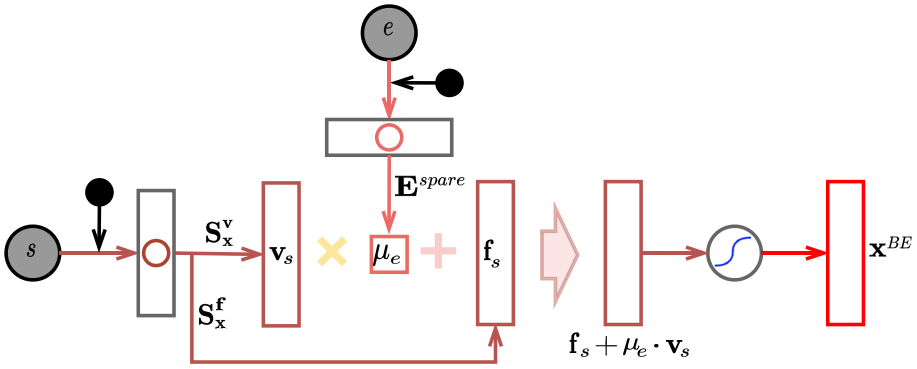


Fig. 4 Representation extraction process of $\mathbf{x}^{BE}$ under the condition of sparse exercise data

parameter of the exercise (refer to Eqs. 1 and 2). Although the setup in RME alleviate over-parameterization and over-fitting of models to a certain, there is a limited amount of information that a 1-dimensional vector can carry compared with a multi-dimensional continuous vector, so the representation of exercises cannot be fully extracted when the exercise data is not sparse. Therefore, RME has been improved to serve as BE in this paper. BE, as the core component, is used to learn the basic representations required for the DLKT model, involving exercise data and skill data.

Let E and S represent the number of distinct exercise tags and distinct skill tags of the e-learning system, respectively. For factor embedding $\mathbf{x}$, when the exercise data is sparse, the exercise tag in each timestamp needs to be scalar embedded to reduce the effect of the exercise data on BE (the relationship between the sparsity of exercise data and its embedding is studied and discussed in Section 6.2.4). The formalization process is as follows:

$$\mu_e = OneHot(e) \cdot \mathbf{E}^{spare}, \tag{3}$$

where $\mathbf{E}^{spare} \in \mathbb{R}^{E\times 1}$ denotes the continuous embedding matrix for any exercise tag e under the exercise data sparsity condition; $\mu_e \in \mathbb{R}$ denotes the scalar embedding of e; $OneHot(\cdot)$ denotes the one-hot encoding operation. The scalar embedding, μ_e, is used to weight the variant vector of the corresponding skill tag s, then the weighted result is added to the factor embedding of s, and finally the sum vector is fed into the *tanh* activation layer to form the corresponding factor embedding of BE (Figure 4 shows the process of representation extraction):

$$\mathbf{x}^{BE} = Tanh(\mathbf{W}_1 \cdot (\mathbf{f}_s + \mu_e \cdot \mathbf{v}_s) + \mathbf{b}_1), \tag{4}$$

where $\mathbf{x}^{BE} \in \mathbb{R}^D$ denotes the factor embedding of BE; $\{\mathbf{W}_1, \mathbf{b}_1\}$ are the corresponding activation layer parameters; $\mathbf{f}_s \in \mathbb{R}^D$ and $\mathbf{v}_s \in \mathbb{R}^D$ denote the factor embedding and variant vector of s respectively, and the formalization processes are as follows:

$$\mathbf{f}_s = OneHot(s) \cdot \mathbf{S}_{\mathbf{x}}^{\mathrm{f}}, \tag{5}$$

$$\mathbf{v}_s = OneHot(s) \cdot \mathbf{S}_{\mathbf{x}}^{\mathrm{v}}, \tag{6}$$

where $\mathbf{S}_{\mathrm{x}}^{\mathbf{f}} \in \mathbb{R}^{S\times D}$ and $\mathbf{S}_{\mathrm{x}}^{\mathbf{v}} \in \mathbb{R}^{S\times D}$ denote the continuous embedding matrices for s. When the exercise data is non-sparse, the factor embeddings of e and s are concatenated and then

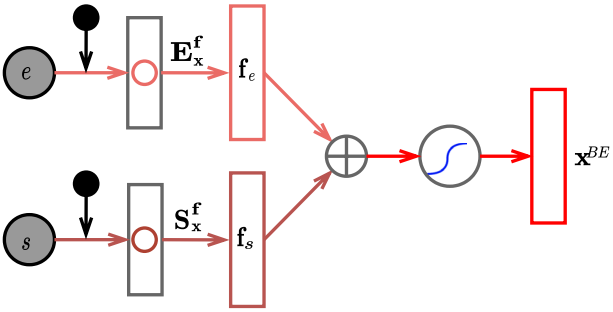


Fig. 5 Representation extraction process of $\mathbf{x}^{BE}$ under the condition of non-sparse exercise data

fed into the *tanh* activation layer to form the corresponding base embedding as (Figure 5 shows the process of representation extraction):

$$\mathbf{x}^{BE} = Tanh(\mathbf{W}_2 \cdot [\mathbf{f}_s \oplus \mathbf{f}_e] + \mathbf{b}_2), \tag{7}$$

where $\{\mathbf{W}_2, \mathbf{b}_2\}$ are the corresponding activation layer parameters; $\oplus$ is the concatenation operation; $\mathbf{f}_e \in \mathbb{R}^D$ denotes the factor embedding of e, and the calculation process is as follows:

$$\mathbf{f}_e = OneHot(e) \cdot \mathbf{E}_{\mathbf{x}}^{\mathbf{f}}, \tag{8}$$

where $\mathbf{E}_{\mathbf{x}}^{\mathbf{f}} \in \mathbb{R}^{E \times D}$ denotes the continuous embedding matrix for any exercise e under the exercise data non-sparsity condition. In summary, the final expression of BE for $\mathbf{x}$ is as follows:

$$\mathbf{x}^{BE} = \begin{cases} Tanh(\mathbf{W}_1 \cdot (\mathbf{f}_s + \mu_e \cdot \mathbf{v}_s) + \mathbf{b}_1) & when \quad sparse \\ Tanh(\mathbf{W}_2 \cdot [\mathbf{f}_s \oplus \mathbf{f}_e] + \mathbf{b}_2) & when \quad non\text{-}sparse \end{cases} \tag{9}$$

For pair embedding $\mathbf{y}$ of BE, $\mathbf{y}^{BE} \in \mathbb{R}^D$ has the same structure as $\mathbf{x}^{BE}$, and the final expression is as follows:

$$\mathbf{y}^{BE} = \begin{cases} Tanh(\mathbf{W}_3 \cdot (\mathbf{p}_{(s,r)} + \mu_e \cdot \mathbf{v}_{(s,r)}) + \mathbf{b}_3) & when \quad sparse \\ Tanh(\mathbf{W}_4 \cdot [\mathbf{p}_{(s,r)} \oplus \mathbf{p}_{(e,r)}] + \mathbf{b}_4) & when \quad non\text{-}sparse \end{cases} \tag{10}$$

where $\{\mathbf{W}_3, \mathbf{b}_3, \mathbf{W}_4, \mathbf{b}_4\}$ are the corresponding activation layer parameters; $\mathbf{p}_{(s,r)} \in \mathbb{R}^D$ and $\mathbf{v}_{(s,r)} \in \mathbb{R}^D$ denote the pair embedding and the variant vector of (s,r), $\mathbf{p}_{(e,r)} \in \mathbb{R}^D$ denote the pair embedding of (e,r), and the calculation processes are as follows:

$$\mathbf{p}_{(s,r)} = MultiHot(s,r) \cdot \mathbf{S}_{\mathbf{y}}^{\mathbf{p}}, \tag{11}$$

$$\mathbf{v}_{(s,r)} = MultiHot(s,r) \cdot \mathbf{S}_{\mathbf{y}}^{\mathbf{v}}, \tag{12}$$

$$\mathbf{p}_{(s,r)} = MultiHot(e,r) \cdot \mathbf{E}_{\mathbf{y}}^{\mathbf{p}}, \tag{13}$$

where $\mathbf{S}_{\mathbf{y}}^{\mathbf{p}} \in \mathbb{R}^{(S+2) \times D}$ and $\mathbf{S}_{\mathbf{y}}^{\mathbf{v}} \in \mathbb{R}^{(S+2) \times D}$ denote the continuous embedding matrices for (s,r); $\mathbf{E}_{\mathbf{y}}^{\mathbf{p}} \in \mathbb{R}^{(S+2) \times D}$ denote the continuous embedding matrix for (s,r); $MultiHot(\cdot)$ denotes the multi-hot encoding operation.

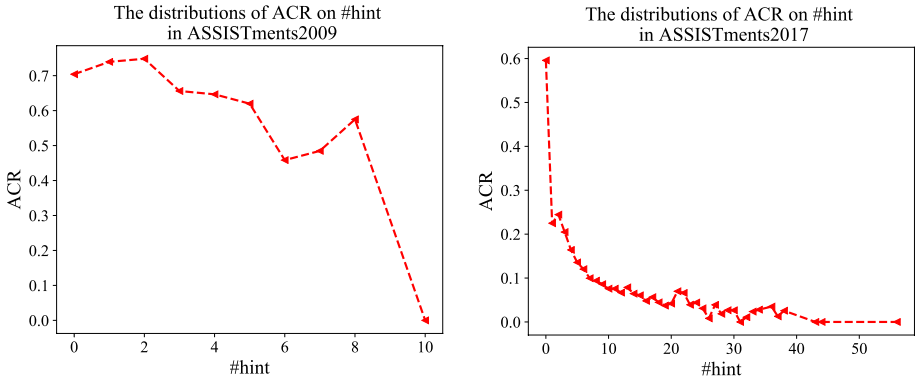


Fig. 6 Distributions of ACR on #hint for datasets ASSISTments2009 and ASSISTments2017 (refer to Table 6)

4.2 Auxiliary embedding

In order to further enrich $\mathbf{x}^{BE}$ and $\mathbf{y}^{BE}$ without over-fitting, we explored the attribute factors of the exercise, which had not been considered in the previous works. Research shows that the attributes of exercise are usually divided into two categories: one is relational attributes (e.g., template, exercise type), the other is indicative attributes (e.g., hint). The former can reflect the relationship between exercises, while the latter can be used as an indicator of the difficulty (or complexity) of the exercise. There are two specific examples to illustrate these two types of attribute factors.

- **Template**. Exercises in e-learning are usually generated based on the template. In other words, multiple exercises may belong to the same template. Compared with the exercise set, the scale of the template set is smaller, but relatively considerable compared with the skill set. Data analysis shows that the difference between the Average Correct Rate (ACR) of exercises under the same template is less than that under the same skill. Therefore, the template information can supply the difficulty difference of the exercise when faced with data sparsity.
- **Hint**. To assist learners in self-learning, some e-learning platforms set up hints for each exercise. Generally, the total number of hints assigned by the platform for each exercise can reflect the difficulty of the exercise to a certain extent. We can see from Figure 6 that the more the number of hints for an exercise, the more difficult the exercise is, given that ACR indicates the difficulty of the exercise.

The factors in different e-learning platforms are different, so the auxiliary factors that can reflect the difficulty of exercise are far from limited to these two types, which inspires us to propose an extensible embedding component, auxiliary embedding (AE), for IR. AE provides BE with embeddings of auxiliary data. Since there are differences in the types and numbers of auxiliary data for different e-learning settings, AE is extensible.

Given N different types of auxiliary factors, and let $\{A_1, A_2, ..., A_N\}$ represent the number of tags for each factor, respectively. For factor embedding $\mathbf{x}$, the one-hot vector of $a_i (i = 1,2,...,N)$ are first embedded by embedding matrices $\mathbf{A}_{\mathbf{x}}^{i} \in \mathbb{R}^{A_i \times D}$ to generate the corresponding factor embeddings, $\mathbf{f}_{a_i} \in \mathbb{R}^{D}$, respectively. The calculation process is as follows:

$$\mathbf{f}_{a_i} = OneHot(a_i) \cdot \mathbf{A}^i_{\mathbf{x}}. \tag{14}$$

Then, all these embeddings are concatenated and then fed into the *tanh* activation layer to generate the factor embedding of AE ($\mathbf{x}^{AE} \in \mathbb{R}^D$) as:

$$\mathbf{x}^{AE} = Tanh(\mathbf{W}_5 \cdot [\mathbf{f}_{a_1} \oplus \mathbf{f}_{a_2} \oplus \cdots \oplus \mathbf{f}_{a_N}] + \mathbf{b}_5), \tag{15}$$

where $\{\mathbf{W}_5,\mathbf{b}_5\}$ are the corresponding activation layer parameters

For pair embedding $\mathbf{y}$, the multi-hot vector of (a_i,r) is first embedded by embedding matrices $\mathbf{A}^i_{\mathbf{y}} \in \mathbb{R}^{A_i \times D}$ to generate the corresponding pair embeddings, $\mathbf{p}_{(a_i,r)} \in \mathbb{R}^D$, respectively. The calculation process is as follows:

$$\mathbf{p}_{(a_i,r)} = MultiHot(a_i, r) \cdot \mathbf{A}^i_{\mathbf{y}}. \tag{16}$$

Then, all these embeddings are concatenated and then fed into the *tanh* activation layer to generate the pair embedding of AE ($\mathbf{y}^{AE} \in \mathbb{R}^D$) as:

$$\mathbf{y}^{AE} = Tanh(\mathbf{W}_6 \cdot [\mathbf{p}_{(a_1,r)} \oplus \mathbf{p}_{(a_2,r)} \oplus \cdots \oplus \mathbf{p}_{(a_N,r)}] + \mathbf{b}_6), \tag{17}$$

where $\{\mathbf{W}_6,\mathbf{b}_6\}$ are the corresponding activation layer parameters.

4.3 Performance embedding

Performance refers to the objective results of the user's past exercise-making behaviors, i.e., the number of correct and incorrect responses in the past. A correct response will help the model to affirm and increase the strength estimate of the user's KS, in the case of current strength is already high. An incorrect response will help users better find the deficiencies of their knowledge reserve. Therefore, incorrect responses may simply lead to more learning than correct responses. However, while making the model sensitive to incorrectness is a good start, it also seems useful to make the model specifically sensitive to correctness.

The performance factor analysis results (as shown in Figure 7) on ASSSISTments2017 and Statics2011 support our motivation. As can be seen that: ARC of learners to the same exercise at the next timestamp gradually increases on the whole, with the increase in the number of historically correct responses; the trend is the opposite for the number of historically incorrect responses. Thus, PE involves the learning of both correctness and incorrectness representations. As shown in Fig. 3, p_{cor} and p_{inc} in PE denote the good and poor performance factors of e in the past respectively. The two delay features are discretized at following scale to alleviate the impact of performance data sparsity:

$$y = log_2(x + 1), \tag{18}$$

where x and y denote the feature values before and after discretization, respectively.

Let P_{cor} and P_{inc} represent the maximum number of correct and incorrect response after discretization, respectively. For factor embedding $\mathbf{x}$, the one-hot vectors of p_{cor} and p_{inc} are used to generate the corresponding factor embeddings, $\mathbf{f}_{p_{cor}} \in \mathbb{R}^D$ and $\mathbf{f}_{p_{inc}} \in \mathbb{R}^D$, respectively. The calculation processes are as follows:

$$\mathbf{f}_{p_{cor}} = OneHot(p_{cor}) \cdot \mathbf{P}^{cor}_{\mathbf{x}}, \tag{19}$$

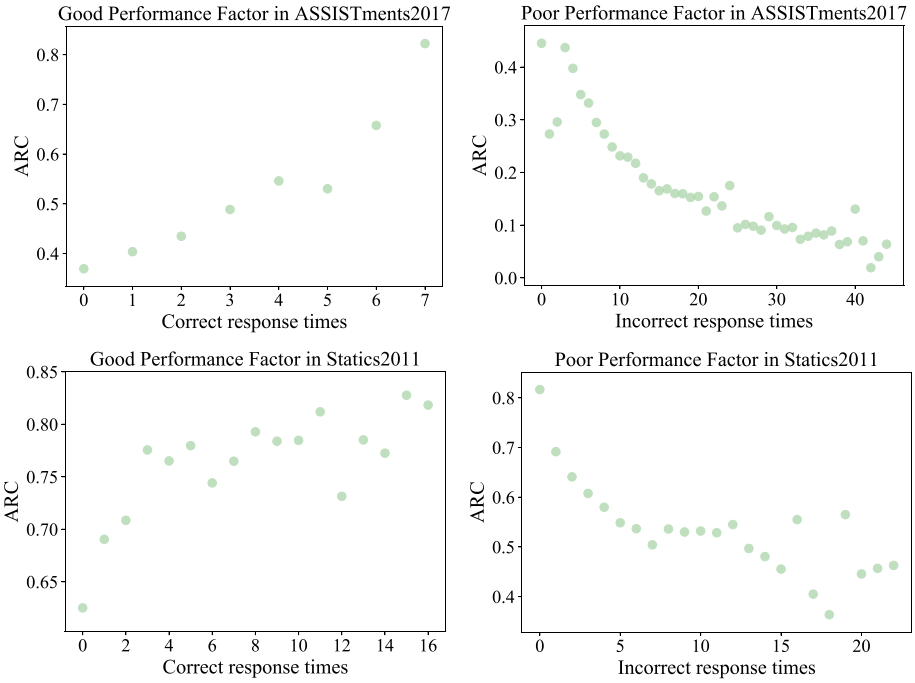


Fig. 7 Distributions of ACR on the number of correct and incorrect response for ASSISTments2017 and Statics2011

$$\mathbf{f}_{p_{inc}} = OneHot(p_{inc}) \cdot \mathbf{P}_{\mathbf{x}}^{inc}. \tag{20}$$

Where $\mathbf{P}_{\mathbf{x}}^{cor} \in \mathbb{R}^{P_{cor} \times D}$ and $\mathbf{P}_{\mathbf{x}}^{inc} \in \mathbb{R}^{P_{inc} \times D}$ denote the continuous embedding matrices for p_{cor} and p_{inc}, respectively. Then $\mathbf{f}_{p_{cor}}$ and $\mathbf{f}_{p_{inc}}$ are concatenated and then fed into the *tanh* activation layer to generate the factor embedding of PE ($\mathbf{x}^{PE} \in \mathbb{R}^D$) as:

$$\mathbf{x}^{PE} = Tanh(\mathbf{W}_7 \cdot [\mathbf{f}_{p_{cor}} \oplus \mathbf{f}_{p_{inc}}] + \mathbf{b}_7), \tag{21}$$

where $\{\mathbf{W}_7, \mathbf{b}_7\}$ are the corresponding activation layer parameters. Figure 8 shows an embedding process for $\mathbf{x}^{PE}$.

For pair embedding $\mathbf{y}$, the multi-hot vectors of (p_{cor},r) and (p_{inc},r) are first embedded by embedding matrices $\mathbf{P}_{\mathbf{y}}^{cor} \in \mathbb{R}^{P_{cor} \times D}$ and $\mathbf{P}_{\mathbf{y}}^{inc} \in \mathbb{R}^{P_{inc} \times D}$ to generate the corresponding pair embeddings $\mathbf{p}_{(p_{cor},r)} \in \mathbb{R}^D$ and $\mathbf{p}_{(p_{inc},r)} \in \mathbb{R}^D$, respectively. The calculation processes are as follows:

$$\mathbf{p}_{(p_{cor},r)} = MultiHot(p_{cor}, r) \cdot \mathbf{P}_{\mathbf{y}}^{cor}, \tag{22}$$

$$\mathbf{p}_{(p_{inc},r)} = MultiHot(p_{inc}, r) \cdot \mathbf{P}_{\mathbf{y}}^{inc}. \tag{23}$$

Then, all these embeddings are concatenated and then fed into the *tanh* activation layer to generate the pair embedding of PE ($\mathbf{y}^{PE} \in \mathbb{R}^D$) as:

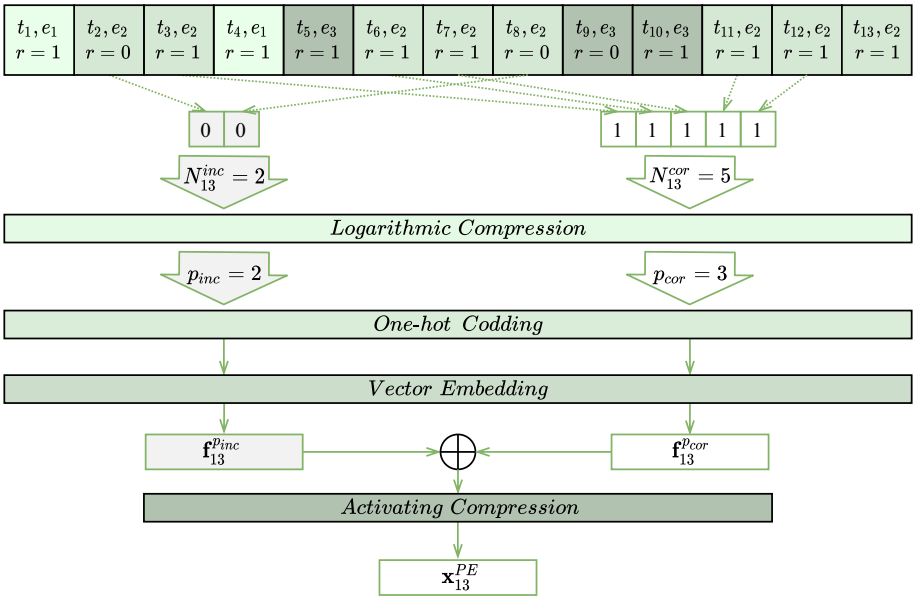


Fig. 8 PE embedding process for factor embedding. Given a learner's historical interaction sequence up to the timestamp 13, generate the corresponding factor embedding of PE

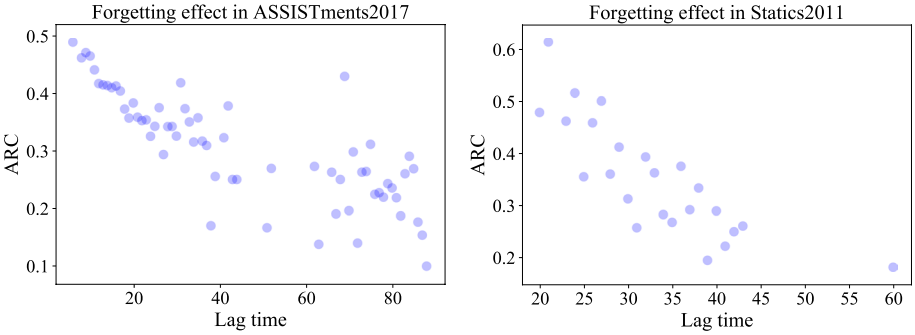


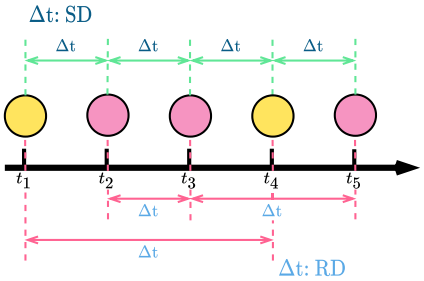
Fig. 9 Correlation between lag time and ARC of exercise in ASSISTments2017 and Statics2011

$$\mathbf{y}^{PE} = Tanh(\mathbf{W}_8 \cdot [\mathbf{p}_{(p_{cor},r)} \oplus \mathbf{p}_{(p_{inc},r)}] + \mathbf{b}_8), \tag{24}$$

where $\{\mathbf{W}_8, \mathbf{b}_8\}$ are the corresponding activation layer parameters.

4.4 Forgetting embedding

Predicting a learner's knowledge precisely is a difficult task because learners do forget, i.e., the time lag (or delay) between the last learning of the same or similar content and the next learning. Nagatani et al. 's research shows that: how the probability of responding correctly depends on the lag time from the previous interaction with the same skill. We analyze the



Timestamp	t_1	t_2	t_3	t_4	t_5
RD	0	0	Δt_{32}	Δt_{41}	Δt_{53}
SD	0	Δt_{21}	Δt_{32}	Δt_{43}	Δt_{54}

Fig. 10 Forgetting factors from a learner's sequence of interactions. Each circle corresponds to an interaction and the same color represents the same exercise id. In the right table, the time gap $\Delta_{ij} = t_i - t_j$

correlation between delay time and ARC of exercise in three datasets: ASSISTments2017 and Statics2011 (Figure 9 shows the analysis results), which further consolidates the above conclusion. As can be seen that: ARC of learners to the same exercise at the next timestamp gradually decreases, with the increase in lag time on the whole.

To achieve an accurate knowledge modeling, we introduce the Forgetting Embedding (FE) component to model the learner's forgetting factors. Different from other work, we consider the following two features in this study:

- Repeated Delay (RD): the time delay between two adjacent interactions with the same exercise id.
- Sequence Delay (SD): the time delay of two successive interactions; the exercise id of an interaction do not matter.

Thus, FE involves the learning of both RD and SD representations. Figure 10 illustrates these delay factor, and the missing RD and SD are set to a fixed value of 0. All the delay features are used by the seconds and are discretized by Eq. 18 to alleviate the impact of delay data sparsity.

Let F_{rep} and F_{seq} represent the maximum time delays of RD and SD after discretization, respectively. For factor embedding $\mathbf{x}$, the one-hot vectors of f_{rep} and f_{seq} are used to generate the corresponding factor embeddings, $\mathbf{f}_{f_{rep}} \in \mathbb{R}^D$ and $\mathbf{f}_{f_{seq}} \in \mathbb{R}^D$, respectively. The calculation processes are as follows:

$$\mathbf{f}_{f_{rep}} = OneHot(f_{rep}) \cdot \mathbf{F}_{\mathbf{x}}^{rep}, \tag{25}$$

$$\mathbf{f}_{f_{seq}} = OneHot(f_{seq}) \cdot \mathbf{F}_{\mathbf{x}}^{seq}. \tag{26}$$

Where $\mathbf{F}_{\mathbf{x}}^{rep} \in \mathbb{R}^{F_{rep} \times D}$ and $\mathbf{F}_{\mathbf{x}}^{seq} \in \mathbb{R}^{F_{seq} \times D}$ denote the continuous embedding matrices for f_{rep} and f_{seq}, respectively. Then $\mathbf{f}_{f_{rep}} \in \mathbb{R}^D$ and $\mathbf{f}_{f_{seq}} \in \mathbb{R}^D$ are concatenated and then fed into the *tanh* activation layer to generate the factor embedding of PE ($\mathbf{x}^{FE} \in \mathbb{R}^D$) as:

$$\mathbf{x}^{FE} = Tanh(\mathbf{W}_9 \cdot [\mathbf{f}_{f_{rep}} \oplus \mathbf{f}_{f_{seq}}] + \mathbf{b}_9), \tag{27}$$

where $\{\mathbf{W}_9, \mathbf{b}_9\}$ are the corresponding activation layer parameters.

For pair embedding $\mathbf{y}$, the multi-hot vectors of (f_{rep}, r) and (f_{seq}, r) are first embedded by embedding matrices $\mathbf{F}_{\mathbf{y}}^{rep} \in \mathbb{R}^{F_{rep} \times D}$ and $\mathbf{F}_{\mathbf{y}}^{seq} \in \mathbb{R}^{F_{seq} \times D}$ denote the continuous embedding

Table 5 Performance evaluation results of ERL+AKT (detailed in Section 5) with different activation functions on datasets: ASSISTments2009, ASSISTments2017 and Statics2011 (detailed in Section 6.1.1)

	ASSISTments2009		ASSISTments2017		Statics2011	
Activation functions	ACC	AUC	ACC	AUC	ACC	AUC
Linear	0.9065	0.8333	0.8756	0.7863	0.8797	0.8170
Softplus	**0.9078**	0.8340	**0.8790**	**0.7897**	**0.8809**	0.8183
ReLU	0.9069	0.8340	0.8751	0.7859	0.8792	0.8172
LeakyReLU	0.9068	0.8340	0.8751	0.7858	0.8793	0.8169
Sigmoid	0.9075	**0.8345**	0.8785	0.7890	0.8802	**0.8185**
Tanh	0.9063	0.8343	0.8756	0.7865	0.8795	0.8166

matrices for f_{rep} and f_{seq} to generate the corresponding pair embeddings $\mathbf{p}_{(f_{rep},r)} \in \mathbb{R}^D$ and $\mathbf{p}_{(f_{seq},r)} \in \mathbb{R}^D$, respectively. The calculation processes are as follows:

$$\mathbf{p}_{(f_{rep},r)} = MultiHot(f_{rep}, r) \cdot \mathbf{F}_{\mathbf{y}}^{rep}, \tag{28}$$

$$\mathbf{p}_{(f_{seq},r)} = MultiHot(f_{seq}, r) \cdot \mathbf{F}_{\mathbf{y}}^{seq}. \tag{29}$$

Then, all these embeddings are concatenated and then fed into the *tanh* activation layer to generate the pair embedding of FE ($\mathbf{y}^{FE} \in \mathbb{R}^D$) as:

$$\mathbf{y}^{FE} = Tanh(\mathbf{W}_{10} \cdot [\mathbf{p}_{(f_{rep},r)} \oplus \mathbf{p}_{(f_{seq},r)}] + \mathbf{b}_{10}), \tag{30}$$

where $\{\mathbf{W}_{10}, \mathbf{b}_{10}\}$ are the corresponding activation layer parameters.

4.5 Embedding integration

In order to provide a model embedding interface for DLKT, the output of the four embedding components needs to be integrated into an embedding vector with fixed dimensions. The most straightforward approach is to concatenate the embedding of the output of these four components as:

$$\mathbf{x}_{concat}^{ERL} = \left[\mathbf{x}^{BE} \oplus \mathbf{x}^{AE} \oplus \mathbf{x}^{PE} \oplus \mathbf{x}^{FE}\right], \tag{31}$$

$$\mathbf{y}_{concat}^{ERL} = \left[\mathbf{y}^{BE} \oplus \mathbf{y}^{AE} \oplus \mathbf{y}^{PE} \oplus \mathbf{y}^{FE}\right]. \tag{32}$$

where $\mathbf{x}_{concat}^{ERL} \in \mathbb{R}^D$ and $\mathbf{y}_{concat}^{ERL} \in \mathbb{R}^D$ denote the factor embedding and pair embedding of ERL based on directly concatenating, respectively.

However, considering the extensibility of the approach, the output dimensions of ERL also need to be fixed when the above four types of information cannot be provided or are not necessary (especially the last three). Therefore, we perform a compression operation on $\mathbf{x}_{concat}^{ERL}$ and $\mathbf{y}_{concat}^{ERL}$. In order to make the compression effect better, we compare the linear activation ("Linear") and five nonlinear activation functions: "Softmax", "Sigmoid", "Tanh", "ReLU" and "LeakyReLU". The results show that Softmax has the best overall performance on the basis of solving the over-fitting (refer to Table 5 and Figure 11, where

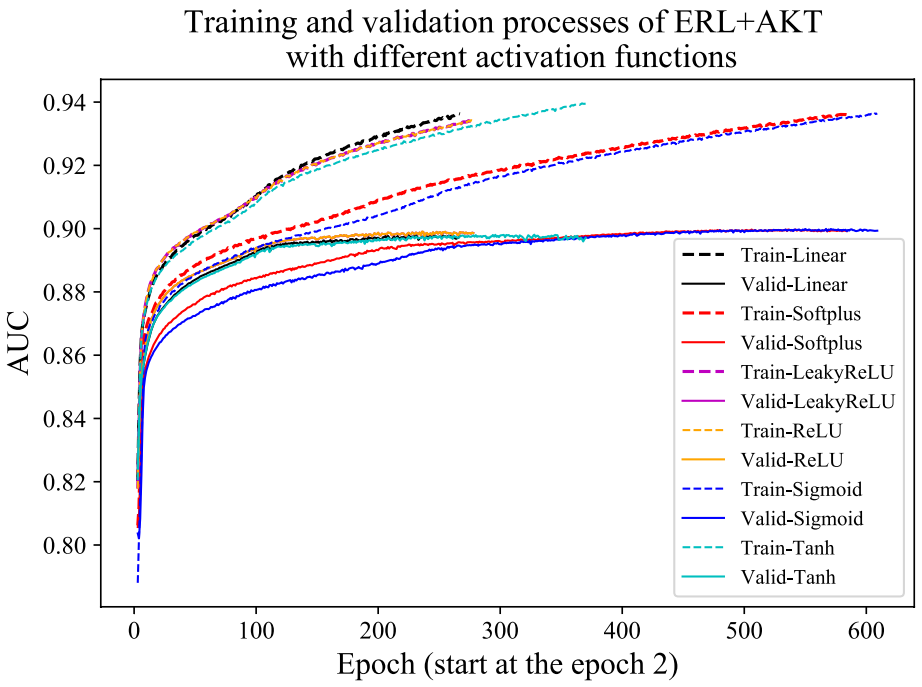


Fig. 11 Training and validation processes of ERL+AKT (detailed in Section 5) with different activation functions on the dataset of ASSISTments2009

ERL+AKT denotes the application of ERL to AKT). Therefore, the final integration form of the proposed ERL approach is as follows:

$$\mathbf{x}^{ERL} = Softplus\big(\mathbf{W}_{11} \cdot \big[\mathbf{x}^{BE}[\oplus\mathbf{x}^{AE}][\oplus\mathbf{x}^{PE}][\oplus\mathbf{x}^{FE}]\big] + \mathbf{b}_{11}\big), \tag{33}$$

$$\mathbf{y}^{ERL} = Softplus\big(\mathbf{W}_{12} \cdot \big[\mathbf{y}^{BE}[\oplus\mathbf{y}^{AE}][\oplus\mathbf{y}^{PE}][\oplus\mathbf{y}^{FE}]\big] + \mathbf{b}_{12}\big), \tag{34}$$

where $\mathbf{x}^{ERL} \in \mathbb{R}^D$ and $\mathbf{y}^{ERL} \in \mathbb{R}^D$ denote the final factor embedding and pair embedding of ERL, respectively; $\{\mathbf{x}^{BE}, \mathbf{y}^{BE}\}$ are required; $\{\mathbf{x}^{AE}, \mathbf{y}^{AE}\}$, $\{\mathbf{x}^{PE}, \mathbf{y}^{PE}\}$, and $\{\mathbf{x}^{FE}, \mathbf{y}^{FE}\}$ are optional; $\{\mathbf{W}_{11},\mathbf{b}_{11}, \mathbf{W}_{12},\mathbf{b}_{12}\}$ are the corresponding activation layer parameters, whose dimensions vary with the number of components to be integrated. In addition, to further avoid over-fitting problems, we add the *drop-out* operation during activation.

5 Applying ERL to DLKT models

In this section, we provide two instances to illustrate how to apply the proposed ERL approach to existing DLKT models. Existing DLKT models are divided into two main classes: RNN-based models and attention mechanism(AM)-based models. For RNN-based models, DKT, as the first application of deep learning in the field of knowledge tracing, uses RNN and LSTM to model knowledge tracing task. MANN extends LSTM and GRU using external memory, and is used by whom to model knowledge tracing tasks. At the

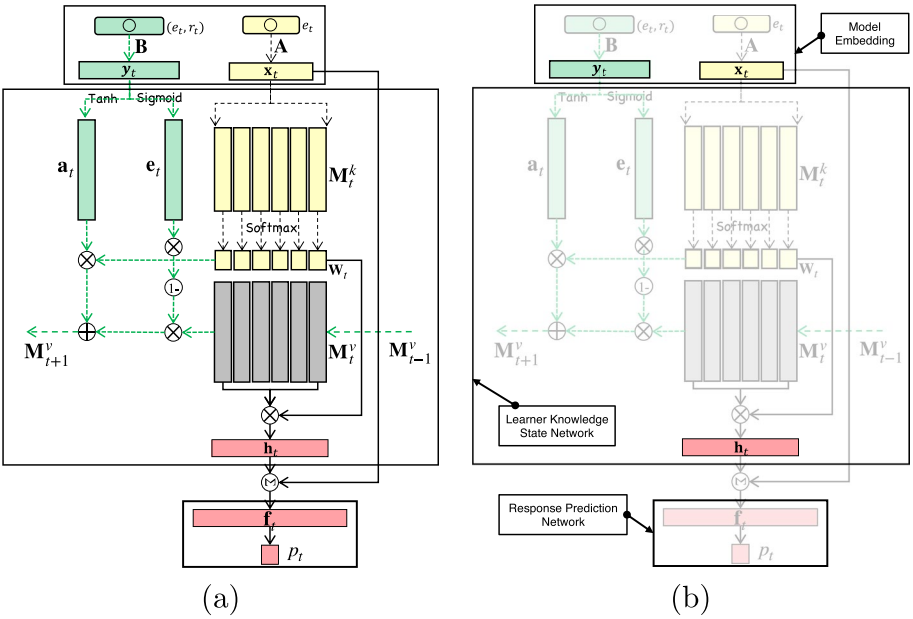


Fig. 12 DKVMN (of (**a**)) and its DLKT-oriented generalized architectures (of (**b**))

same time, they propose DKVMN based on MANN, taking into account the correlation between skills in the knowledge tracing field. For AM-based models, SAKT, as the first proposed model, aims to deal with the sparse problem of exercise data, which is a self-attention based knowledge tracing model. RKT extends SAKT by introducing a relation-aware self-attention layer that incorporates the contextual information. AKT, a completely dependent on attention network, extends SAKT by building context-aware representations of exercises and responses and proposing a monotonic attention mechanism to summarize past learner performance in the right time scale. To sum up, we therefore choose DKVMN and AKT as the application models of ERL. We select two existing main stream DLKT models for improved instances. The first instance improves DKVMN by ERL, named ERL+DKVMN; the second improves AKT by ERL, named ERL+AKT.

5.1 Application Instance-1: ERL+DKVMN

Figure 12(a) shows the knowledge tracing process of DKVMN. At the timestamp t, DKVMN traces the KS of the learner by reading and writing to the *value*-memory matrix $\mathbf{M}_t^v$ using the correlation weight computed from the input skill and the *key*-memory matrix $\mathbf{M}^k$, and predicts the response of the learner to the skill based on the read memory content $\mathbf{r}_t$ and the input skill embedding $\mathbf{k}_t^s$. $\mathbf{M}^k$ and $\mathbf{M}^v$ are used to store the underlying concepts and the mastery levels of each concept, respectively.

According to the architecture of DLKT, DKVMN can be generalized into three parts: *Model Embedding*, *Learner Knowledge State Network* and *Response Prediction Network*, as shown in Figure 12(b). ERL+DKVMN is produced by extending *Model Embedding* in DKVMN with the proposed ERL representation learning approach,

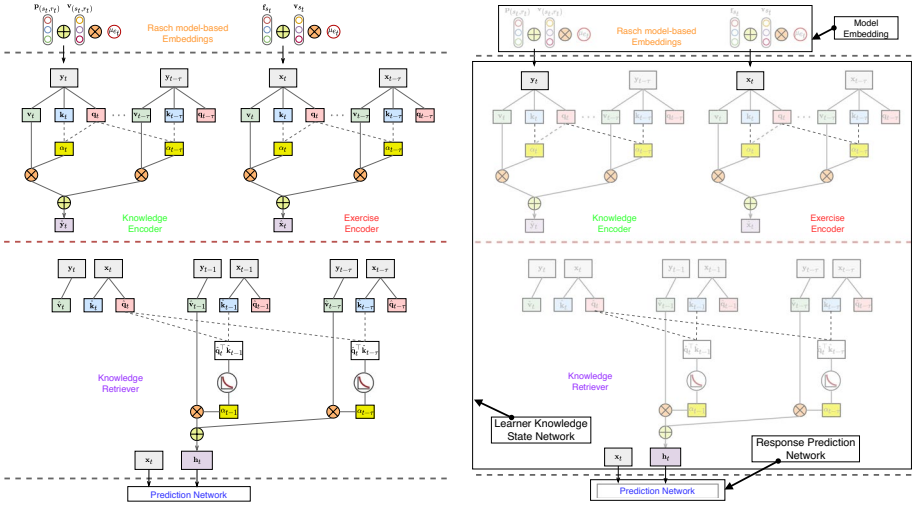


Fig. 13 AKT (of (**a**)) and its AKT-oriented generalized architectures (of (**b**))

using both factor embedding ($\mathbf{x}^{ERL}$) and pair embedding $\mathbf{y}^{ERL}$ of ERL to improve the original factor embedding ($\mathbf{x}^{DKVMN} = \mathbf{f}_s$) and pair embedding ($\mathbf{y}^{DKVMN} = \mathbf{p}_s$) in *Model Embedding* of DKVMN.

5.2 Application Instance-2: ERL+AKT

AKT [11] is the representative work based on attention mechanism, which consists of five components: *Rasch Model-based Embeddings*, *Question Encoder*, *Knowledge Encoder*, *Knowledge Retriever* and *Prediction Network*, as shown in Figure 13(a). *Rasch Model-based Embeddings* is used as raw embeddings for exercises and responses; *Question Encoder* and *Knowledge Encoder* are used to compute the context-aware representations of exercises and responses pairs, respectively; *Knowledge Encoder* uses these representations as input and computes the KS of the learner; *Prediction Network* is used to predict the learner's response to the current exercise. According to the architecture of DLKT, *Rasch Model-based Embeddings* corresponds to *Model Embedding*; *Question Encoder*, *Knowledge Encoder* and *Knowledge Retriever* correspond to *Learner Knowledge State Network*; *Prediction Network* corresponds to *Response Prediction Network*. The generalized architectures for AKT is shown in Figure 13(b).

ERL+AKT is produced by extending *Model Embedding* in AKT with the proposed ERL representation learning approach, using both factor embedding ($\mathbf{x}^{ERL}$) and pair embedding $\mathbf{y}^{ERL}$ of ERL to improve the original factor embedding ($\mathbf{x}^{DKVMN} = \mathbf{f}_s$) and pair embedding ($\mathbf{y}^{DKVMN} = \mathbf{p}_s$) in *Model Embedding* of AKT.

Table 6 Statistical information for all datasets

Datasets	#users	#skills	#exercises	#interactions	Seqlen[a]	Sparsity[c]
ASSISTments2009	4217	124	26688	525534	[1, 8214]	99.53%
ASSISTments2017	1709	102	3162	942816	[2, 3057]	82.55%
Statics2011	333	300	1224	261947	[5, 2185]	35.73%

[a]"Seqlen" denotes the range of sequence lengths

[b]"Sparsity" is calculated as $Sparsity = 1 - \#interactions/(\#users \cdot \#exercises)$

5.3 Model extensibility and training

For application models of ERL, *local* extensibility means that one or more auxiliary factors can be extended in AE to meet different KT tasks with various auxiliary factors, and you can also mine more performance and forgetting factors to enrich PE and FE, respectively. In addition to the *local* extensibility, these application instances also show *global* extensibility in terms of the overall structure, which means that all the *local* embedding components other than BE can be used separately or not, and you can also extend more user-related *local* embedding components other than PE and FE to enrich UE, respectively.

All parameters in ERL and its variants are learned together with parameters of the DLKT model from downstream All learnable parameters in the entire are trained in end-to-end fashion by minimizing the following cross entropy loss between predicted response (r'_t) and real response (r_t) during training:

$$\mathcal{L} = -\sum_t \left(r_t \log\left(r'_t\right) + \left(1 - r_t\right) \log\left(1 - r'_t\right)\right), \tag{35}$$

where $\mathscr{L}$ denotes the cross entropy loss.

6 Experiments

To evaluate the usability and effectiveness of the proposed representation learning framework, we apply the proposed ERL into two representative existing DLKT model: DKVMN and AKT (the latest DLKT network), and the two applied instances respectively are denoted by ERL+DKVMN and ERL+AKT. We design six experiments on four real-world datasets to comprehensively evaluate ERL+DKVMN and ERL+AKT.

6.1 Experimental settings

6.1.1 Datasets

Three real-word benchmark datasets are used to evaluate the performance of all the models involving in experiments on predicting future learner responses, including ASSISTments2009, ASSISTments2017 and Statics2011. For ASSISTments2009, each exercise involves the specific skill, template which is used to generated related exercises, and hint number which is set by the platform according to the specific exercise. For

ASSISTments2017, each exercise involves the specific skill, hint number and exercise types. For Statics2011, each exercise involves the specific steps. Previous works use exercise tag and step tag together to retrieve each record of exercise-making. The original exercise tag is treated as the skill tag, and the original exercise tag is treated as a new exercise tag along with the step tag in the paper. The complete statistical information for all datasets refers to Table 6. We delete user sequences of length 1 from all the datasets, and round off all responses. To ensure the integrity of the learning sequence, we save all records of interactions missing skill (all missing skills are treated as a new skill label) in ASSISTments2009, which is different for all existing works. We trained all the models with 80% of the dataset and test them on the remaining. We perform 5-fold cross validation to evaluate all the models on all datasets, in which folds are split based on users.

6.1.2 Auxiliary factor selection

Different e-learning platforms have different types and numbers of exercise attributes, so it is crucial to determine which auxiliary factors contribute to the final performance of the model. This paper uses the method of *analysis* + *statistics* + *experiment*:

- *Analysis*. Potential auxiliary factors that may affect the difficulty of exercise are firstly selected from the existing auxiliary factors by analysis, such as the generated template of exercise and the number of hints equipped with exercise. For the template, the difficulty of the exercise is greatly influenced by the template, e.g. "1 + 1" (exercise) for "$a + b$" (template), "$1 - (2 + 3)$" (exercise) for "$a - (b + c)$" (template); For the hint, the more hints an exercise is equipped with, the more difficult it may be intuitively.
- *Statistics*. The uncertain auxiliary factors (e.g. hints) can be further determined by means of data statistics (refer to Section 4.2), in which the ACR of exercises can be used as an indicator of whether the corresponding factors can affect the difficulty of exercises.
- *Experiment*. The final judgment of whether a potential auxiliary factor should be used to trace the knowledge state of learners, it is necessary to verify whether its integration can improve the evaluation metrics of KT task through specific experiments.

6.1.3 Model setting and evaluating

Except for all the scalar parameters (with the same dimension of 1), all the vector embeddings in DKVMN and ERL+DKVMN have the same dimension of 100, and all the vector embeddings in AKT and ERL+AKT have the same dimension of 256; the dropout rate for all models is set to 0.05. The Area Under the Curve (AUC) and Accuracy (ACC) are used to evaluate the performances of all the models on predicting binary-valued future user responses to exercises. Generally, the value 0.5 of AUC or ACC represents the performance prediction result by randomly guessing, and the larger the better.

6.1.4 Factor encoding

All the input factors are presented to neural networks using "one-hot" encoding vectors. Take exercise factor, for example, if E different exercise tags exist in total, then the "one-hot" encoding of the exercise tag e_t is length E vector whose entries are all zero except that the e_t^{th} entry is one. All the input pairs are presented using "multi-hot" encoding vectors.

Table 7 Examples for the input encoding

Exercise tag	Response	Factor encoding	Pair encoding
Exercise-1	Correct (or 1)	[1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1]
Exercise-1	Incorrect (or 0)	[1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0]
Exercise-2	Correct (or 1)	[0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 1]
Exercise-3	Incorrect (or 0)	[0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0]
Exercise-4	Correct (or 1)	[0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 1]
Exercise-5	Incorrect(or 0)	[0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0]

Specifically, the "one-hot" encoding of the exercise tag e_t is directly concatenated with the response of e_t to form the "multi-hot" encoding vectors of the pair (e_t,r_t). An concrete example of the input encoding for exercise is illustrated in Table 7 where there is a total of five exercises. An alternative encoding for the pair encoding is "one-hot" encoding, whose vector is twice as long as the "multi-hot" encoding. In our experiments, we have found that using the "multi-hot" encoding for pair is much more effective and introduces fewer model parameters.

6.2 Experimental design and results

In this section, we design four sub-experiments to answer the following research questions (RQs):

- RQ1: Can ERL improve the DLKT networks?
- RQ2: What is the effect of various components in ERL?
- RQ3: What is the effect of various factors in each *local* embedding component of ERL?
- RQ4: How to determine when to use scalar or vector embedding for exercise data?

6.2.1 Overall performance evaluation (RQ1)

To evaluate the usability and effectiveness of the proposed representation learning approach, We compare our models involving with the state-of-the-art DLKT methods. The details of compared models are:

- **DKT** [30] is the earliest DLKT method that leverages single layer LSTM to model learner knowledge state.
- **DKT+** [37] is an improved version of DKT with regularization on prediction consistency, which reconstructs the observed input and overcomes the prediction performance inconsistency of model across time-steps.
- **DKVMN** [38] is improved memory augmented recurrent neural network with dynamic key-value memories, which mines correlations between skills.
- **SAKT** [26] applies the transformer structure to assign weights to the previously learned exercises for predicting predict the learner's response to the current exercise.
- **AKT** [11] is an improved version of SAKT with contextualized representations of exercises and responses, which utilizes a monotonic attention mechanism to summarize past learner performance, and the RME model to capture individual differences among exercises covering the same skill.

Table 8 Performance comparison before and after application of ERL

	ASSISTments2009		ASSISTments2017		Statics2011	
Models	AUC	ACC	AUC	ACC	AUC	ACC
DKVMN	0.8206	0.7699	0.7232	0.6911	0.8164	0.7741
ERL+DKVMN	0.8857	0.8077	0.8613	0.7725	0.8591	0.8039
Improved	7.93%	4.91%	19.10%	11.78%	5.23%	3.85%
AKT	0.8608	0.8000	0.7664	0.7149	0.8151	0.7776
ERL+AKT	0.9078	0.8340	0.8753	0.7862	0.8924	0.8299
Improved	5.46%	4.25%	14.21%	9.97%	9.48%	6.73%

Table 9 AUC performance of other baseline DLKT methods on all datasets on predicting future learner responses

Datasets	DKT	DKT+	DKVMN	SAKT	AKT	ERL+ DKVMN	ERL+ AKT
ASSISTments-2009	0.8170	0.8024	0.8206	0.7520	0.8608	0.8857	0.9078
ASSISTments-2017	0.7264	0.7124	0.7232	0.6569	0.7664	0.8613	0.8753
Statics-2011	0.8233	0.8301	0.8164	0.8029	0.8151	0.8591	0.8924

Where we re-implement DKVMN and AKT in PyTorch, and the rest of the experimental results are replicated from AKT because the data are pre-processed and the parameters are initialized in exactly the same way.

we firstly compare the performance of DKVMN [38] and AKT [11] before and after the application of ERL, and Table 8 shows the performance of all DLKT methods across all datasets on predicting future learner responses. The results show that ERL can effectively improve the performance of the DLKT models on predicting future learner responses. Compared with original DKVMN, the improved predictive performances of ERL+DKVMN evaluated by AUC are up to 7.93%, 19.10% and 5.23% on datasets ASSISTments2009, ASSISTments2017 and Statics2011, respectively; the improved predictive performances evaluated by ACC are up to 4.91%, 11.78% and 3.85% on datasets ASSISTments2009, ASSISTments2017 and Statics2011, respectively. Compared with original AKT, the improved predictive performances of ERL+AKT evaluated by AUC are up to 5.46%, 14.21% and 9.48% on datasets ASSISTments2009, ASSISTments2017 and Statics2011, respectively; the improved predictive performances evaluated by ACC are up to 4.25%, 9.97% and 6.73% on datasets ASSISTments2009, ASSISTments2017 and Statics2011, respectively. This experiment demonstrates that ERL can greatly improve the performances of the DLKT models on predicting future learner responses.

In addition, we compare the AUC performance of ERL+DKVMN and ERL+AKT with other DLKT model, and Table 9 shows the compared results. It can be seen that AKT shows better performance than other baseline models on all data except for Statics2011; DKVMN has better performance than other RNN-based models (DKT and DKT+) on the whole; the proposed ERL+AKT shows the best performance over all involved baseline models. This experiment demonstrates that ERL-enhanced AKT achieves the best performance across all involved benchmark datasets on predicting future learner responses. Although data enhancement leads to the introduction of more parameters into the model, our series of specific operations (such as scalar embedding

Table 10 Global ablation results of ERL, where "–" means the corresponding item is missing due to the absence of the corresponding factors in Statics2011

Datasets	BE	AE	PE	FE	AUC (DKVMN)	ACC (DKVMN)	AUC (AKT)	ACC (AKT)
ASSISTments2009					0.8206	0.7699	0.8608	0.8000
	✓				0.8326	0.7740	0.8628	0.8004
	✓	✓			0.8402	0.7724	0.8743	0.8083
	✓		✓		0.8827	0.8078	0.8966	0.8244
	✓			✓	0.8391	0.7787	0.8642	0.8004
	✓	✓	✓	✓	0.8858	0.8077	0.8989	0.8277
ASSISTments2017					0.7232	0.6911	0.7664	0.7149
	✓				0.7457	0.7018	0.7799	0.7210
	✓	✓			0.8473	0.7620	0.8675	0.7799
	✓		✓		0.7671	0.7151	0.8134	0.7494
	✓			✓	0.7544	0.7075	0.7880	0.7280
	✓	✓	✓	✓	0.8613	0.7724	0.8689	0.7803
Statics2011		–			0.8164	0.7741	0.8151	0.7776
	✓	–			0.8536	0.7996	0.8726	0.8111
	✓	–	✓		0.8581	0.8021	0.8841	0.8221
	✓	–		✓	0.8433	0.7929	0.8790	0.8163
	✓	–	✓	✓	0.8591	0.8039	0.8924	0.8299

of exercise labels, logarithmic discretization of integer data, etc.) have in fact reduced the influence to a certain extent and reached a fully acceptable degree.

6.2.2 Global ablation study (RQ2)

To get deep insights on ERL, we investigate the contribution of various components involved in ERL to the whole performance. Therefore, we conduct some ablation experiments to show how each embedding component of ERL affect final results. All the datasets are used to support the *global* ablation study of ERL, and Table 10 shows the ablation results, where AE is not integrated in ERL due to lack of time data for Statics2011. It can be seen from Table 10 that i) the ERL integrating all the embedding components improves the performance of AKT more than the other variants; ii) ERL with BE and AE improves the performance of AKT more than ERL with BE and PE or FE on ASSISTments2017; iii) ERL with BE and PE improves the performance of AKT more than ERL with BE and AE or FE on ASSISTments2009 and Statics2011; iv) ERL with BE and FE improves the performance of AKT less than ERL with BE and AE or PE on all datasets; v) the variant of ERL only integrating the embedding component of BE outperforms the corresponding base models. In conclusion, the effectiveness of ERL comes from all the embedding components working together, different embedding components have different positive effects on the whole performance of models, and no one embedding component is significantly better than any other on all datasets.

Table 11 Local ablation results of AE, PE and FE

Components	Variants	ASSISTments2009		ASSISTments2017		Statics2011	
		AUC	ACC	AUC	ACC	AUC	ACC
Baseline	BE	0.8206	0.7699	0.7232	0.6911	0.8151	0.7776
AE	BE+a_1	0.8736	**0.8085**	0.7878	0.7273	–	–
	BE+a_2	0.8609	0.7991	**0.8677**	0.7794	–	–
	BE+AE	0.8743	0.8083	0.8675	0.7800	–	–
PE	EB+p_{cor}	0.8853	0.8132	0.7825	0.7253	0.8831	0.8210
	BE+p_{inc}	0.8927	0.8237	0.7996	0.7387	0.8790	0.8166
	BE+PE	0.8967	0.8244	0.8134	0.7494	0.8841	0.8221
FE	EB+f_{seq}	0.8636	0.7979	0.7803	0.7225	0.8783	**0.8168**
	BE+f_{rep}	**0.8650**	**0.8006**	0.7812	0.7229	0.8774	0.8165
	BE+FE	0.8642	0.8005	0.7850	0.7254	0.8790	0.8163

6.2.3 Local ablation study (RQ2)

To get deep insights on each embedding component in ERL, we conduct some ablation experiments to investigate the contribution of each factor in AE, PE and FE to the *local* performances based on BE. Table 11 shows the *local* ablation results of AE, PE and FE.

For AE, two datasets with two auxiliary factors are used to support the *local* ablation study of AE, where a_1 and a_2 denote the template and hint factors (for ASSISTments2009), the type and hint factors (for ASSISTments2017), respectively. It can be seen from Table 11 that the variant of ERL (BE+AE) integrating two auxiliary factors based on BE improves the performance of AKT more than the variants of ERL integrating a single auxiliary factor, and the variant of ERL only involving BE performs the worst. In addition, different auxiliary factors have different positive effects on model performance. In conclusion, the more effective factors are integrated in the model embedding, the better the performance of the model can be significantly improved.

For PE, all the datasets are used to support the *local* ablation study of PE. Where BE+p_{cor} and BE+p_{inc} denote the variants of ERL integrating the correct and incorrect response factor, respectively. It can be seen from Table 11 that the variant of ERL (BE+PE) integrating complete performance factors based on BE improves the performance of AKT more than the variants of ERL integrating a single performance factor, and the variant of ERL only involving BE performs the worst. In addition, although different performance factors have different positive effects on model performance, BE+p_{inc} achieves a larger performance improvement than BE+p_{cor} on the whole, which also supports our point in Section 4.3. In conclusion, the integration of performance factors in the model embedding can effectively improve the model performance under different e-learning settings.

For FE, all datasets with time data are used to support the *local* ablation study of FE. Where BE+f_{seq} and BE+f_{rep} denote the variants of ERL integrating the *sequence delay* and *repeat delay* factors, respectively. It can be seen from Table 11 that the variant of ERL (BE+FE) integrating both forgetting factors based BE improves the performance of AKT more than the variants of ERL integrating a single forgetting factor on the whole, and the variant of ERL only involving BE performs the worst. In addition, different forgetting factors have different positive effects on model performance. In conclusion, the integration of

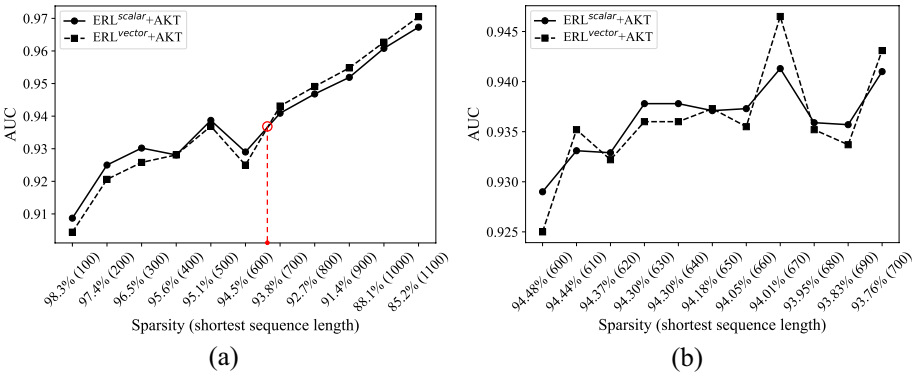


Fig. 14 Comparative study between the scalar and vector embeddings. Where (**a**) shows the result that the shortest sequence length is in the interval 100 to 1100, and the step size is 100; (**b**) shows the result in the interval 600 to 700, and the step size is 10

forgetting factors in the model embedding can effectively improve the model performance under different e-learning settings with time data.

Above all, although the variant of ERL integrating two factors from any one embedding component improves the performance of AKT more than the variants of ERL integrating one factor on the whole, sometimes the variants integrating one factor also shows better performance than the variants integrating two factor, such as: ACC for AE on ASSISTmens2009, AUC for AE on ASSISTmens2017, AUC and ACC for FE on ASSISTmens2009, and ACC for FE on Statics2011. Therefore, this indicates that our model still has deficiencies in the integration of factors inside embedding components, which will be one of the directions of our future efforts.

6.2.4 Exercise embedding study (RQ4)

In the Section 4.1, it is stipulated that when the exercise data is sparse, scalar embedding is carried out for the exercise tag; otherwise, vector embedding is carried out. However, sparse or not is a fuzzy concept, which can not provide specific guidance for practical application. Therefore, in this section, we make a comparative study of the scalar embedding and vector embedding under different exercise sparsity conditions based on the applied instance ERL+AKT.

The sparse dataset, ASSISTments2009, is first divided into a series of sub-datasets with different sparsity according to the shortest sequence length. Then, the performance of the ERL+AKT model based on scalar embedding and vector embedding is respectively evaluated on all sub-datasets (let ERL^{scalar}+AKT and ERL^{vector}+AKT denote the scalar embedding-based and vector embedding-based models, respectively). Figure 14 shows results of the comparative study in the interval $100 \sim 1100$. In our experiments, we have found that if the shortest sequence length in the sub-dataset is less than 100, the AUC value of the scalar embedding is always greater than that of the vector embedding; the opposite is true if the shortest sequence length is greater than 1100.

As you can see from Figure 14(a), results show that: (i) as the sparsity of exercise data decreases, the performance of ERL^{scalar}+AKT and ERL^{vector}+AKT increases on the whole, which is consistent with the intuition; (ii) when the sparsity is relatively large, the

performance of ERL^{scalar}+AKT is better than that of ERL^{vector}+AKT; otherwise, the performance of ERL^{vector}+AKT is better; (iii) the red circle is the intersection point of the two polylines, and the corresponding sparsity lies between 94.5% and 93.8%. In order to further explore more accurate sparsity, we conducted more fine-grained exploration in the range of $600 \sim 700$. As you can see from Figure 14(b), results show that: although the overall trend of the two polylines is in line with expectations, they intersect several times. Therefore, we cannot obtain a more accurate sparsity boundary between the scalar embedding and the vector embedding more accurately. In conclusion, we suggest that when the data sparsity is relatively large, the scalar embedding should be used for ERL; when the data sparsity is relatively small, the vector embedding is used for ERL. In other cases, the vector embedding is used, because we find in the experiment that the scalar embedding requires more time to train the model than the vector embedding.

7 Conclusion and future works

In this paper, we find that the mining and integration of learning-related factors can effectively improve the performance of DLKT models by analyzing previous studies. However, due to the difference of learning content and learning environment in different e-learning, types and quantities of learning-related factors modeled in the specific models are different, which is not conducive to the subsequent application and promotion of the models. We focus on providing a model embedding interface for DLKT by consider multiple types of learning-related factors.

Starting from the nature of learning behavior and combining with previous research, we first explore and analyze four types of learning-related factors: exercise and skills, attributes of exercise, learners' historical performance, and learners' forgetting behavior in the learning process. An Extensible Representation Learning (ERL) approach for DLKT is then proposed to extract and integrate the representations of the four types of factors by setting five components. Finally, we apply ERL into two mainstream DLKT models, and results on three real-world datasets show that the proposed approach can significantly improve performances of DLKT models on predicting future learner responses. In the future, our work will focus on the application of ERL on more DLKT models, mining and itegrating more factors related to item and user to rich the representation learning model, designing more effective factor integration networks to give full play to the role of all factors.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments. The research is supported by the National Natural Science Foundation of China (61702532, 61690203).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, pp 17–36 (2012)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)

3. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
4. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis–a general method for cognitive model evaluation and improvement. In: International Conference on Intelligent Tutoring Systems, pp 164–175 (2016)
5. Chaudhry, R., Singh, H., Dogga, P., Saini, S.K.: Modeling Hint-Taking behavior and knowledge state of students with Multi-Task learning. Int. Educ. Data Mining Soc. (2018)
6. Cinquin, P.A., Guitton, P., Sauzéon, H.: Online e-learning and cognitive disabilities: a systematic review. Comput. Educ. **130**, 152–167 (2019)
7. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Model. User-adapted Interact. **4**(4), 253–278 (1994)
8. Dauphin, G.M.Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Bergstra, J, et al.: Unsupervised and transfer learning challenge: a deep learning approach. In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, pp 97–110 (2012)
9. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Murphy, K., Strohmann, T., Sun, S., Zhang, W., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 601–610 (2014)
10. Dong, G., Zhang, X., Lan, L., Wang, S., Luo, Z.: Label guided correlation hashing for large-scale cross-modal retrieval. Multimed. Tools Appl. (2019)
11. Ghosh, A., Heffernan, N., Lan, A. S.: Context-aware attentive knowledge tracing. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 2330–2339 (2020)
12. He, L.: Integrating performance and side factors into embeddings for deep Learning-Based knowledge tracing. In: 2021 IEEE International Conference on Multimedia and Expo (ICME) (2021)
13. He, L., Tang, J., Li, X., Wang, T.: ADKT: Adaptive deep knowledge tracing. In: International Conference on Web Information Systems Engineering, pp. 302–314 (2020)
14. Hinton, G.E.: Learning distributed representations of concepts. In: Proceedings of the Eighth Annual Conference of the Cognitive Science Society, vol. 1, p 12 (1986)
15. Khajah, M.M., Huang, Y., González-Brenes, J.P., Mozer, M.C., Brusilovsky, P.: Integrating knowledge tracing and item response theory: a tale of two frameworks. CEUR Workshop Proc. **1181**, 7–15 (2014)
16. Khajah, M., Lindsey, R.V., Mozer, M.C.: How deep is knowledge tracing?, arXiv:1604.02416 (2016)
17. Krishnan, R., Singh, J., Sato, M., Zhang, Q., Ohkuma, T: Incorporating wide context information for deep knowledge tracing using attentional bi-interaction (2021)
18. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
19. Liu, Y., Hua, W., Qu, J., Xin, K., Zhou, X.: Temporal knowledge completion with context-aware embeddings. World Wide Web **24**(2), 675–695 (2021)
20. Liu, Q., Huang, Z., Yin, Y., Chen, E., Xiong, H., Su, Y., Hu, G.: Ekt: Exercise-aware knowledge tracing for student performance prediction. IEEE Trans. Knowl. Data Eng. **33**(1), 100–115 (2019)
21. Liu, K., Liu, W., Ma, H., Huang, W., Dong, X.: Generalized zero-shot learning for action recognition with web-scale video data. World Wide Web **22**(2), 807–824 (2019)
22. Liu, T., Pan, X., Wang, X., Feenstra, K.A., Heringa, J., Huang, Z.: Predicting the relationships between gut microbiota and mental disorders with knowledge graphs. Health Inf. Sci. Syst. **9**(1), 1–9 (2021)
23. Liu, T., Pan, X., Wang, X., Feenstra, K.A., Huang, Z.: Exploring the Microbiota-Gut-Brain axis for mental disorders with knowledge graphs. J. Artif. Intell. Med. Sci. (2020)
24. Nagatani, K., Zhang, Q., Sato, M., Chen, Y.Y., Chen, F., Ohkuma, T.: Augmenting knowledge tracing by considering forgetting behavior. In: The World Wide Web Conference, pp. 3101–3107 (2019)
25. Niu, L., Fu, C., Yang, Q., Li, Z., Chen, Z., Liu, Q., Zheng, K.: Open-world knowledge graph completion with multiple interaction attention. World Wide Web **24**(1), 419–439 (2021)
26. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. In: Proceedings of the 12th International Conference on Educational Data Mining, pp 384–389 (2019)
27. Pandey, S., Srivastava, J.: RKT: Relation-aware self-attention for knowledge tracing. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp 1205–1214 (2020)
28. Pavlik, P.I. Jr, Cen, H., Koedinger, K.R: Performance factors analysis–A new alternative to knowledge tracing. Online submission (2009)
29. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Long short-term memory. Neural Comput. **8**(9), 1735–1780 (1997)

30. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., Sohl-Dickstein, J.: Deep knowledge tracing. Adv. Neural Inf. Process. Syst., 505–513 (2015)
31. Rollinson, J., Emma, B.: From Predictive models to instructional policies. Int. Educ. Data Mining Soc. (2015)
32. Wang, Z., Li, L., Zeng, D.: Knowledge-enhanced natural language inference based on knowledge graphs. In: Proceedings of the 28th International Conference on Computational Linguistics (2020)
33. Wilson, K. H., Karklin, Y., Han, B., Ekanadham, C.: Back to the basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation. arXiv:1604.02336 (2016)
34. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Proceedings of the AAAI Conference on Artificial Intelligence, 30(1) (2016)
35. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Deep hierarchical knowledge tracing. In: Proceedings of the 12th International Conference on Educational Data Mining (2019)
36. Yeung, C.K.: Deep-IRT: Make deep learning based knowledge tracing explainable using item response theory. arXiv:1904.11738 (2019)
37. Yeung, C.K., Yeung, D.Y.: Addressing two problems in deep knowledge tracing via Prediction-Consistent regularization. In: Proceedings of the Fifth Annual ACM Conference on Learning at Scale (2018)
38. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th International Conference on World Wide Web, pp 765–774 (2017)
39. Zhang, L., Xiong, X., Zhao, S., Botelho, A., Heffernan, N.T.: Incorporating rich features into deep knowledge tracing. In: Proceedings of the Fourth ACM Conference on Learning@scale, pp 169–172 (2017)
40. Zhang, M., Zhu, J., Wang, Z., Chen, Y.: Providing personalized learning guidance in MOOCs by multi-source data analysis. World Wide Web **22** (3), 1189–1219 (2019)

Authors and Affiliations

Liangliang He[1] · Jintao Tang[1] · Xiao Li[2] · Pancheng Wang[1] · Feng Chen[1] · Ting Wang[1]

Liangliang He
heliangliang19@nudt.edu.cn

Jintao Tang
tangjintao@nudt.edu.cn

Xiao Li
xiaoli@nudt.edu.cn

Pancheng Wang
wangpancheng13@nudt.edu.cn

Feng Chen
chenfeng15a@nudt.edu.cn

[1] College of Computer, National University of Defense Technology, Yanwachi Street, Changsha 410073, Hunan, People's Republic of China

[2] Information center, National University of Defense Technology, Yanwachi Street, Changsha 410073, Hunan, People's Republic of China